

1982 APL Users Meeting

Proceedings Volume I **Applications Sessions**



Sponsored by:
I.P. Sharp Associates Limited

1982 APL Users Meeting

Proceedings Volume I
Applications Sessions

I.P. Sharp Associates Limited
Toronto, Ontario
1982

PROCEEDINGS: VOLUME I

APPLICATIONS SESSIONS

LIST OF PAPERS BY SESSION

ACTUARIAL AND INSURANCE APPLICATIONS

1. Mooney, Gary C.
Actuaries, Models and APL 65
2. Bauer, Michal I.
Developing APL Systems at The Urbaine Life Re 248
3. Kaye, Randy
Valuation Concepts in APL 8
4. Fitzpatrick, Tim P.
APL Beyond the Actuarial Department 325

BUDGETING APPLICATIONS

1. Elliott, Maurice
ABS—A Budget System 32
2. Ford, David
Government Budgeting: A Flexible Solution to a Large Problem 254
3. Palese, Vincent J., Jr.
Top Down Budgeting: A Fair-Share Approach 48
4. Ford, Alan C.
Comprehensive Corporate Budgeting—Planning, Tracking and Commitment Control with APL 135

EXPLOITING PUBLIC DATA BASES

1. Sheer, Jonathan
Monitoring the U.S. Economy Using Citibank Data Bases 395
2. Keith, David
Let the Numbers Do the Talking—with MAGIC 155
3. Boulay, Richard A.
A Toolkit for Time Series Operations 230
4. Mikelson, Peter G.
Uses of International Environmental Data *

FINANCIAL APPLICATIONS

1. Besse, Robert A.
Adapting a Financial Planning System 21
2. Lefebvre, Denis
APL as a Tool for Financial Applications 338
3. Peckham, Lawrence, and Donald Soule
The GO: A Comprehensive, Integrated Financial Reporting
System in APL *
4. Peterson, Terry B.
A Modelling Tool for Evaluating Proposed Oil and Gas
Exploration Expenditures 115
5. Parker, Graham
Running a Money Market System on SHARP APL 330

INTERNATIONAL BANKING APPLICATIONS

1. Fuller, Mark
Monitoring of International Exposures 241
2. DePrince, Albert E., Jr.
APL Support of Economic Research: Costs and Benefits *
3. Dobrzanski, Christopher
Country Risk Analysis: A Quantitative Approach to
the Bank of British Columbia 125
4. Crouse, David W.
Support Systems for Asset-Liability Management 171

MANAGEMENT INFORMATION SYSTEMS

1. Waller, Michael J.
A Case Study: APL as the 40% Solution 143
2. Stahlbaum, B.W., and A.H. Meek
The Design and Implementation of an Integrated Computer
System in a Veterinary College 56
3. Isobe, Hiroshi, and Kotaro Yamashita
EXCEED—Executive Management Decision Support System 296

MANAGEMENT SCIENCE APPLICATIONS

1. Jenkins, Rodman
Petroleum Reservoir Simulation in APL 283
2. Willems, Alain
Using the Box and Jenkins Method for the Analysis of
European Air Traffic 364
3. Boyd, E.A.
Interactive Decision Making: Creating Decision Trees
with APL 100
4. Lemberg, Ray
APL in Risk Analysis 13

MULTINATIONAL FINANCIAL SYSTEMS

1. Brunsman, K.K., and Ed E. Irsch
Financial Consolidations with Timesharing 335
2. Courtney, Thomas P., and Paul MacLauchlan
Moore Corporation's Financial Consolidation System 264
3. Steinitz, G.B.
Management Accounting in International Banking 75

PRODUCTION PLANNING AND CONTROL SYSTEMS

1. Corrie, Walter John
The PROMIS Mine Planning and Control System 91
2. Andersen, Henning Lervad
APL as a Tool in Chemical Production 147
3. Tenalio, Michael J.
Tracking and Measuring the Forecast 272

PROGRAMMING TOOLS AND TECHNIQUES

1. McGrew, Jon
Using Defined Operators 342
2. McDonnell, Eugene
APL as a Functional Programming Language 108

- | | |
|--|-----|
| 3. Ryan, James L.
A Rationalization of the Syntax and Semantics of
APL Expressions | * |
| 4. Sykes, Roy A., Jr.
Managing APL Projects | 387 |

THE APL ARENA: AVE

- | | |
|--|-----|
| 1. Montalbano, Michael
A Personal History of APL | 192 |
| 2. Moore, Roger
Network Management Tools | 222 |
| 3. Iverson, Eric
Current Developments in SHARP APL | 83 |
| 4. Jordan, Ray
The Current and Future Markets for APL | 1 |

THE APL ARENA: VALE

- | | |
|---|-----|
| 1. Polivka, Ray P.
Current SIGAPL Activities | 260 |
| 2. Foster, Garth H.
Standards Update—Will it Ever Mean Anything? | * |
| 3. McIntyre, Donald
APL | * |

* Papers unavailable at time of publication

LIST OF APPLICATION PAPERS BY AUTHOR

Andersen, Henning Lervad	APL as a Tool in Chemical Production	147
Bauer, Michal I.	Developing APL Systems at The Urbaine Life Re	248
Besse, Robert A.	Adapting a Financial Planning System	21
Boulay, Richard A.	A Toolkit for Time Series Operations	230
Boyd, E.A.	Interactive Decision Making: Creating Decision Trees with APL	100
Brunsmann, K.K., and Ed E. Irsch	Financial Consolidations with Timesharing	355
Corrie, Walter John	The PROMIS Mine Planning and Control System	91
Courtney, Thomas P., and Paul MacLauchlan	Moore Corporation's Financial Consolidation System	264
Crouse, David W.	Support Systems for Asset-Liability Management	171
DePrince, Albert E., Jr.	APL Support of Economic Research: Costs and Benefits	*
Dobrzanski, Christopher	Country Risk Analysis: A Quantitative Approach to the Bank of British Columbia	125
Elliott, Maurice	ABS—A Budget System	32
Fitzpatrick, Tim P.	APL Beyond the Actuarial Department	325
Ford, Alan C.	Comprehensive Corporate Budgeting—Planning, Tracking and Commitment Control with APL	135

Ford, David	Government Budgeting: A Flexible Solution to a Large Problem	254
Foster, Garth H.	Standards Update—Will it Ever Mean Anything?	*
Fuller, Mark	Monitoring of International Exposures	241
Irsch, Ed E., and K.K. Brunsman	Financial Consolidations with Timesharing	355
Isobe, Hiroshi, and Kotaro Yamashita	EXCEED—Executive Management Decision Support System	296
Iverson, Eric	Current Developments in SHARP APL	83
Jenkins, Rodman	Petroleum Reservoir Simulation in APL	283
Jordan, Ray	The Current and Future Markets for APL	1
Kaye, Randy	Valuation Concepts in APL	8
Keith, David	Let the Numbers Do the Talking—with MAGIC	155
Lefebvre, Denis	APL as a Tool for Financial Applications	338
Lemberg, Ray	APL in Risk Analysis	13
Maclauchlan, Paul, and Thomas P. Courtney	Moore Corporation's Financial Consolidation System	264
McDonnell, Eugene	APL as a Functional Programming Language	108
McGrew, Jon	Using Defined Operators	342
McIntyre, Donald	APL	*

Meek, A.H., and B.W. Stahlbaum	The Design and Implementation of an Integrated Computer System in a Veterinary College	56
Mikelson, Peter G.	Uses of International Environmental Data	*
Montalbano, Michael	A Personal History of APL	192
Mooney, Gary C.	Actuaries, Models and APL	65
Moore, Roger	Network Management Tools	222
Palese, Vincent J., Jr.	Top Down Budgeting: A Fair-Share Approach	48
Parker, Graham	Running a Money Market System on SHARP APL	330
Peckham, Lawrence, and Donald Soule	The GO: A Comprehensive, Integrated Financial Reporting System in APL	*
Peterson, Terry B.	A Modelling Tool for Evaluating Proposed Oil and Gas Exploration Expenditures	115
Polivka, Ray P.	Current SIGAPL Activities	260
Ryan, James L.	A Rationalization of the Syntax and Semantics of APL Expressions	*
Sheer, Jonathan	Monitoring the U.S. Economy Using Citibank Data Bases	395
Soule, Donald, and Lawrence Peckham	The GO: A Comprehensive, Integrated Financial Reporting System in APL	*

Stahlbaum, B.W., and A.H. Meek	The Design and Implementation of an Integrated Computer System in a Veterinary College	56
Steinitz, G.B.	Management Accounting in International Banking	75
Sykes, Roy A., Jr.	Managing APL Projects	387
Tenaglio, Michael J.	Tracking and Measuring the Forecast	272
Waller, Michael J.	A Case Study: APL as the 40% Solution	143
Willems, Alain	Using the Box and Jenkins Method for the Analysis of European Air Traffic	364
Yamashita, Kotaro, and Hiroshi Isobe	EXCEED—Executive Management Decision Support System	296

THE CURRENT AND FUTURE MARKETS FOR APL

Ray Jordan
Editor
APL Market News
Mount Carmel, Connecticut

Introduction

There was a time when describing the APL market was an easy task conceptually. You simply counted the number of big IBM computers running APL, and the number of accounts on timesharing services running APL on big IBM computers. This was the market. In the intervening decade-and-a-half, however, the APL market has broadened, branched, rooted and splintered itself in ways that defy a single, simplistic description of today's market. The current APL market shows several faces, each in its own unique phase of growth.

What is APL market?

In many respects, the market for APL is the sum of the markets for all of the application areas which APL supports. Thus, for I.P. Sharp Associates, demand for APL is linked in a very real economic sense to demand for any of the proprietary data bases which are accessed by APL. An APL consultancy which specializes in utility planning systems sees the market for APL expanding or contracting just as the utility markets grow or wane. This illustrates the fact that versions of APL (like most programming languages and computer hardware) are really commercial distribution systems, not end products. No one really wants an expensive blue steel box for their company, worse yet one that generates Greek characters. What the market seeks are applications, and if they come bundled with blue boxes, white typewriters, or programmers who type in Greek, so be it.

I can't discuss all the prospects for each of the application areas which APL serves, but I'd like to illustrate the prospects for APL itself. Towards this end, I think of the commercial market for APL in terms of the machines on which it runs, and the people who know how to run it on those machines. This allows me to compare the language against itself and against alternative programming languages to achieve some measurement of growth.

Before I discuss the specific APL marketing environments, I'd like to sketch a quick, broad vista of commercial APL.

First, there are over 100,000 people who know some APL. As evidence of this, over

90,000 copies of the Gilman & Rose introductory textbook on APL published by Wiley & Sons have been distributed since late 1976. (And these figures represent only the second edition of the work, called *APL: An Interactive Approach*.) Distribution of the book over its lifespan has not shown significant increases nor decreases by year, except for a slight trail-off in recent years, due most likely to competing titles and the growing age of the title. Using this as a guideline, I judge that the growth in the base of APL-educated people has been essentially linear over the last few years.

The growth in the number of companies promoting APL products has been far from linear, however. In an earlier article [Ref.1], I pointed out the impressive growth in the number of APL vendors from an estimated 10 in the late 1960s and early 1970s, to 40 in the middle and later 1970s, to 80 at about 1980. In the intervening two years since 1980, my count of indentified APL vendors has risen dramatically to about 140, far from a linear increase. The APL vendor base now includes most major computer and peripheral manufacturers.

Using 140 as the number of companies currently vending APL-related products or services, and using information gained from surveys on a subset of this group [Ref. 2], I estimate that the equivalent of approximately 3,500 full-time people are engaged in the creation, sale and distribution of APL goods and services. This excludes persons merely using the language. Figuring that each person employed in APL generates gross revenues in the \$50,000 to \$60,000 range annually, this gives APL an estimated current market in the neighborhood of \$200 million annually. Again, this figure excludes the customers' costs of employing people who use APL, which would be some multiple higher than the above figure.

Although this gives an overview of the current APL market, understanding the dynamics of the market requires a closer look at each of the specific markets which comprise APL.

Timesharing

The early commercial thrust in APL was provided almost exclusively by the pioneering timesharing vendors I.P. Sharp Associates from a base in Canada, and STSC, Inc., from the United States, in conjunction with earlier software work by IBM. Since the early 1970s numerous other vendors have added APL to their timesharing offerings. The growth of APL in the timesharing markets has followed a fairly classical product marketing life cycle over its first decade and a half.

The classical phases of product life [Ref.3] show a product moving through the following sequence: *introduction*—characterized by little competition, extensive efforts to convince prospects that this type of product is worth trying, relatively low volume distribution; *growth*—characterized by rising sales and profits, competition, increased number of purchase outlets, beginning of product differentiation and segmentation of market; *maturity*—characterized by decrease in rate of sales increase, decline in margin of profits, price competition, and possible efforts at revitalization; *saturation*—characterized by leveling of sales and increased efforts to make replacement sales; *decline*—characterized by cost controls and abandoning of market by some competitors.

The APL timesharing business has moved through the stages of introductory product growth into the stages of growth and maturity.

During the introductory phase of the early 1970s, APL timesharing vendors were aggressively "spreading the word", convincing this and that manager, executive or scientist to give APL a try. The world was an oyster for raw APL. All business was new business. Sales grew. Profits grew. Branch offices sprouted. APL companies went public. All training was free. The sense was that, if I could show you how good APL was, you would use my APL.

With the growth of the market and the increased access to APL by the middle 1970s, competition developed. Other full-service timesharing vendors offered APL. Benchmarks appeared. Price competition began. Serious APL vendors began to differentiate their products with formatters, keyed-file systems, proprietary data bases, international networks and other languages. APL timesharing vendors sought segmented markets of people like financial analysts, actuaries, airline planners and utility managers to use special APL software.

Today, salesmen may not so much be saying, "Here, **buy** APL", as, "Here, **buy my** APL." To the extent this is happening, it marks the arrival of a mature product. The recent purchase offer of STSC by Continental Telecom after some months of poor profitability lends further evidence that APL timesharing may be entering a rather mature stage. Enclosed, or nested, arrays from a purely marketing standpoint, may be seen as an effort to revitalize a mature product. This is a common market occurrence, sometimes very effective.

The future for APL in the timesharing areas lies in the cost-effective presentation of the language through tools oriented to specific user groups. As Ian Sharp said last year in an interview [Ref. 4], "Software is our product. We're not against selling raw APL, but there's not much of a market in that."

In comparison with other languages, APL in the timesharing business represented quite a marketing coup. No other programming language with which I am familiar was successful in recruiting such a following of timesharing vendors to devote such a level of exclusive commitment. In the early 1970's, because of APL's precedence as a high-level interactive language, the slice of the overall timesharing business devoted to APL was significantly higher than its overall representation on in-house systems. In terms of the hardware supporting APL, virtually every major timesharing company has offered APL, although they may not actively support, enhance or market it now. Also, a significant number of corporate purchasers or users of timesharing have been aware of the language as an option for some time.

Large computers

The trajectory of APL in the large computer market (non-desktop computers) has lagged the timesharing path by a few years. This is partly because APL timesharing use motivates later purchase of an in-house APL machine. Also, the marketing of APL has not been as avidly embraced by large computer manufacturers as it was by earlier timesharing vendors.

Nonetheless, various sources claim a 50% annual growth in the number of VSAPL licenses assigned by IBM, now believed to be in the neighborhood of 3,000 to 5,000 licenses total. Some companies, including I.P. Sharp Associates, STSC and Interprocess Systems as well as IBM, have been busy recently in the system software aftermarket for companies which have installed mainframes with APL capability.

I judge APL on large computers to have just entered its growth phase. If so, we should be seeing rising profits and competition, and the beginnings of special APL software offerings to more segmented markets in the large systems area.

According to a 1981 study by International Data Corp.[Ref.5], APL was in use for between one and three percent of new programming efforts at data processing sites with 20 or more programmers. Since COBOL, FORTRAN and ASSEMBLER were seen as locking up over 75% of these new efforts, considerable room for growth in use of the APL language still exists on mainframes.

Most managers of data processing departments have by now heard of APL, although they may not be well-versed in its use and application. Promotional and educational activities to bring APL's applicability in front of corporate data processing management should fuel growth in this market area. These promotional efforts may be seen in IBM's Information Center concept and in Sharp's in-house APL course. Coverage of the language in such publications as *Computerworld* and James Martin's book, *Application Development Without Programmers* will further reinforce management recognition of the language.

Market factors constraining APL in this market are likely to include:

- 1) adverse initial reaction to the character set;
- 2) objections to the rich expressive capability of the APL syntax on grounds of poor readability and maintainability; and
- 3) preference for high-level languages whose syntax looks more like written English.

Factors which favour the growth of APL in the large system market include:

- 1) dropping machine and storage costs, which require that less penalties be paid for an interpretive language fond of working memory;
- 2) growing cost and scarcity of programmers, which places a premium on highly productive languages like APL;
- 3) availability of the APL character set on terminals and printers from most manufacturers, with gradually fading cost penalties; and
- 4) APL familiarity obtained by managers moving into user companies from IBM, APL timesharing vendor companies, and business schools which offer APL training.

Although APL on mainframes has clearly entered the stage of growth, the slope and duration of the growth curve remain to be defined by the current mix of APL products and corporate needs. A good percentage of machines support the language in this market, but still a relatively small percentage of people using the machines are familiar with APL.

Small computers

The microcomputer market for APL has been in existence for over a decade, since the release of an APL microcomputer by MCM Computers in 1971. MCM's recent demise does not imply that this market is in its declining stage. In fact, most indications are that this product market is still quite early in its introduction. Only about a dozen companies are working the market [Ref. 6] with very few microcomputers having more than one version of APL available (IBM's personal computer is a notable exception).

Although APL now runs under most current microcomputer operating systems, less than one-tenth of one percent of existing microcomputer units are estimated to be running APL. If we assume that APL should be at least as attractive to home and business microcomputer owners as it is to large companies (i.e., 1% to 3% use), then the microcomputer APL market should be able to grow from 15 to 30 times its current level, even without an increase in the underlying installed base of microcomputers.

One key to this level of introductory growth is emphasis on marketing and advertising. Microcomputer-related advertising in *APL Market News* has recently caught up with large system and timesharing advertising in volume. The promotional method in these early stages will be, "Here, try APL on your microcomputer", and not, "Here's why you should buy my APL for your microcomputer". This introductory level of marketing will continue until most of the sellers and prospective buyers of microcomputers have been exposed to APL as a language option for their machine(s).

Costs of implementing APL will be a more significant factor in this market than they are in the large computer market. The cost of hardware to implement the APL character set plus the cost of a sufficient amount of memory to support APL are now an insignificant part of a mainframe budget. But if you take a micro system that costs \$3,000, then add \$200 for an APL character generator, plus \$200 for APL software, plus \$200 for an extra 64K of memory, you have just added 20% to the cost of a system. Twenty percent may be little penalty to an APL'er with conviction, but it is a steep wager for a beginner who has never before heard of the language.

The microcomputer market is also proving itself somewhat resistant to programming languages *per se*, probably because most micro users do not have sufficient time nor incentive to program. What language is VISICALC written in? The truth is, it does not matter. If you can build functional software for a microcomputer user in APL, that user will run APL if he or she wants the software.

On the positive side, IBM's release of its personal computer is helping to legitimize use of the microcomputer in the corporate environment. This environment is the one which has historically been most receptive to APL. Adoption of APL by users and programmers in this environment may promote its spread into other areas of microcomputer use, such as small business and home.

The bundling of Waterloo's version of APL into the Commodore microcomputer was a means of distributing APL without the usual cost penalty associated with the language. Bundling of the language into a top-selling machine may prove to be the most effective mechanism for building the microcomputer market for APL.

Also encouraging has been the vigorous acceptance of APL by non-English speaking countries. The largest single shipment of APL books from the APL Market Place this year went to a microcomputer store in Japan. The avowed intent of Japanese computer

hardware manufacturers to provide low-priced, high-performance micros, coupled with their need to fill a widening software gap, will encourage their close evaluation of APL.

Still in its early stages here, APL has approached only a tiny portion of the microcomputer market. Few machines and relatively few people associated with this market are acquainted with APL.

Other markets

There are several other commercial sub-markets based on APL. Most of these are secondary or parasitic in nature, in that their existence depends upon the fortunes or misfortunes of the three primary markets described above. Still, each of the sub-markets serves to spur growth in the primary markets for APL.

APL peripherals, including screens, terminals, and printers, form a substantial secondary market, brought about because of APL's unique character set. This market is mature in that a majority of larger peripheral manufacturers now support the APL character set on one or more of their machines. This achievement in the peripherals sub-market has removed much of the resistance concerning lack of access to the APL character set from the mainframe and microcomputer markets. The peripheral APL market will continue growing as the sum of the three primary markets.

APL consulting depends for its market primarily upon access to APL through large computers and timesharing. The number of APL consultants continues to increase as the timesharing business is nearing maturity, because the mainframe APL business has been in a growth phase. Microcomputer markets will add some demand for APL consulting, but not at the volume of the larger systems markets.

APL application software has generally been provided by APL vendors themselves, or by consultants with a software package in their back pocket. Keen potential for application or utility software to spur APL sales exists, particularly in the microcomputer market. Application and utility software has already created much APL demand in the timesharing and large computer markets.

Education and training have nurtured the use of APL in its timesharing and large system markets. Both I.P. Sharp Associates and Numetrix have marketed specific course offerings to in-house APL users. Conferences sponsored by Sharp, STSC and the ACM are other reflections of this service. No significant educational offerings targeted specifically to the microcomputer APL market have appeared as of this writing.

Like education, APL-related publications depend upon, as well as promote, the primary APL markets. Both *APL Quote Quad* and *APL Market News* circulate to avid subscribers, and some free vendor publications are available. *APL Market Place* lists over a dozen current APL book titles, and many APL software and hardware user manuals which are commercially available. The book market for APL is likely to grow as APL grows into the microcomputer markets, due to the opportunity for high potential sales volume for books in this market.

Another growing APL sub-market is that of personnel agents specializing in APL recruitment. I know of four agencies in this category currently, up from only one last year. This sub-market is directly linked, of course, to APL's performance in the large computer environment.

Even more unusual sub-markets have been seen sprouting around the APL world lately, such as APL cartoonists, APL joke editors and APL musicians, but I will put off an in-depth look at these until another time.

Summary

APL products and services are estimated to be a \$200 million market annually, in different stages of growth. The timesharing portion of the business is mature, with most timesharing vendors and most timesharing prospective users aware of the language. The large computer market for APL is into a growth stage, with most large computers supporting the language, but a relatively small percentage of potential users using it. The microcomputer market for APL is in its introduction, with little competition for any given microprocessor, and only a tiny percentage of prospective users familiar with the language. Various sub-markets for APL goods and services both depend upon and influence the success of these three primary markets.

References

1. *Computerworld*. February 15, 1982.
2. Information taken from a survey in preparation for an APL Market Directory. APL companies with under 100 employees averaged about 8 employees each. Figuring 8 employees for each of 130 APL companies with under 100 employees yields 1,040 employees. Estimate 2,500 employees directly engaged commercially in APL at timesharing vendors and computer manufacturers selling APL.
3. Stanton, William. *Fundamentals of Marketing*. New York: McGraw Hill, 1978.
4. *APL Market News*, Second quarter 1981, Box 5314, Mt Carmel, Ct. 06518.
5. Personal communication from William Zachmann, Research Director, International Data Corp., Framingham, MA.
6. Cherlin, Mokurai. *APL Micro Report*. Distributed by APL Market Place, Mt. Carmel, CT.

VALUATION CONCEPTS IN APL

Randy Kaye
MONY Life of Canada
Toronto, Ontario

Major valuations in APL? You've got to be kidding! It can't be done: it's too expensive to run and too complicated to maintain compared to COBOL or FORTRAN. Besides, the organizational problems are horrendous.

This paper attempts to show that the generation of large files of reserve factors **can** be done, and for a reasonable cost. Furthermore, when properly organized, the problem falls apart, making trivial the final task of program development. And a by-product of straight-forward program development is the ease of maintenance of those programs.

This approach will be demonstrated through simple examples of individual net level premium reserves. However, I don't view this as a restriction on the general applicability of these techniques.

Preliminary concepts

Usual computational approaches to the task of reserve factor generation often mask the distinction between assumptions and formulae. We need to distinguish the interest and mortality assumptions (and whatever others may be required) from the formulae which will put them together. Each formula is tied to a type of plan for which reserves need to be calculated. Since there will probably be far fewer sets of assumptions than plan types and since these assumption sets will optimally be totally independent of plan type, worthwhile economies can be realized.

We will pre-calculate and file the assumption sets to be used later. The filing away of the assumption sets is greatly facilitated through the use of the SHARP APL data type known as a package. Several variables can be packaged together and filed as one entity. Each assumption set is a calculation environment which will be defined in the workspace before a formula is executed. Since there are fewer assumption sets than plan types, we will leave each calculation environment in place until all reserves using it are calculated.

There are some generalizations we can make regarding the formulae, whatever the plan types are. Prospective reserve formulae are often the easiest to write and we will use them here. Each formula can be broken down to its constituent elements: an insurance,

less a valuation premium, times an annuity. And each of these elements can be expressed using commutation functions.

Assumption sets

The packaged calculation environments contain whatever commutation functions and utility variables are required by any of the formulae using the assumption set. We can standardize these as a set of variables, for example: I, V, D, N , and M , but you may want to include other commutation functions and control variables for your specialty plans. Some may feel that commutation functions are an anachronism from the days before computers, but the computational speed and simplifications resulting from them are no less applicable today, when the power and speed of a computer can be brought to bear. They are flexible for fashioning formulae and their extensive use in actuarial work makes them a powerful tool for communicating actuarial concepts.

Each assumption set may be one of two distinct forms:

- 1) Select, used where there is
 - a) select mortality and varying interest,
 - b) select mortality and level interest, or
 - c) ultimate mortality and varying interest, and
- 2) Ultimate, used where there is ultimate mortality and level interest.

The complexities of the select method need not be reflected in the formulae. We include in the packaged calculation environment another variable called *INDEX*. This variable is used in initializing the calculation environment to conform the variables indexing the commutation vectors to the proper format.

For the details of this select and ultimate indexing technique and an efficient utility for calculating the commutation vectors, please refer to the seminar notes of I.P. Sharp's course entitled "Advanced APL Coding Techniques for Actuarial Applications".

Formulae

Each reserve formula is made up of three factors: an insurance, a valuation premium and an annuity. Each of these factors has, in turn, several formula types. It is our intention to independently develop APL programs for all formula types of these factors and then to combine them according to the requirements of the particular plan types. For example, the insurance factors which we will need include whole life, endowment, level term, decreasing term, and the one or two specialty plans that every company has. Extended use may be made of some of these formula types, for clearly the Whole Life insurance factor will also be used in valuing the Life at 65 benefit. We will not write a separate (duplicate) formula for both. Annuity factors have less variety in practice, but we do not restrict the possibility of an unorthodox one being required. The valuation premium factor can generally be expressed as the "at issue" case of the insurance and annuity factors. However, it will always be distinctly and explicitly calculated.

The APL programs for each of these factors should have as parameters the set of ages and durations required to be calculated. In general, the reserves for all ages and

durations can be calculated in a loop-free manner, regardless of the complexity of the benefit or parameters.

Organization and program development

We set up a control matrix to be accessed by the driver function. Each row denotes a particular plan type's calculation requirements. The fields defined by the columns are as follows:

Field 1:	Plan Code
Field 2:	Low Issue Age
Field 3:	High Issue Age
Field 4:	Low Duration
Field 5:	High Duration
Field 6:	Benefit Period
Field 7:	Benefit Period Age vs Duration Flag
Field 8:	Premium Period
Field 9:	Premium Period Age vs Duration Flag
Field 10:	Report 1—Assumption Set Number
Field 11:	Report 1—Reserve Formula Number
Field 12:	Report 2—Assumption Set Number
Field 13:	Report 2—Reserve Formula Number
...	...

For each government or other report required we loop through the rows of the matrix, processing each plan type in turn. Before we make each pass of the loop, however, we sort the rows of the matrix according to that report's Assumption Set Number. In this way we don't overwrite a calculation environment until we have used it as much as we can.

This main loop in the driver function reads, in turn, a row of the control matrix and defines the following variables:

AGES - based on fields 2 and 3,
DRNS - based on fields 4 and 5,
BP - based on fields 6 and 7, and
PP - based on fields 8 and 9.

Then it calls the function *REPORT N* (where *N* is the report number) to calculate the applicable reserves. Next it calls the function *SAVE N* to save the reserve factors in a file.

We can best discuss the function *REPORT* by showing it;

```

▽ REPORT N;X;XT
[1] GETΔASSUMPTIONΔSET SETΔNUMBER N
[2] X←AGES PLUS 0 ◇ XT←AGES PLUS DRNS
[3] Δ'RES',FORMULAΔNUMBER N
▽

```

The function *GETΔASSUMPTIONΔSET* reads the Assumption Set Number passed to it from the function *SETΔNUMBER* and globally defines the packaged calculation environment specified, from the file of assumption sets. However, if the assumption set re-

quired is already in the workspace, *GETASSUMPTIONSET* will not redefine it. The function *PLUS* uses the calculation environment variable *INDEX* to properly set up the variables *X* and *XT* (i.e., *x* and *x + t*) for indexing the commutation vectors from the assumption set. Finally, a reserve function is executed, specified by the Reserve Formula Number passed to it from the function *FORMULANUMBER*.

Let us look at one such possible reserve function, *RES01*:

```

      ▽ RES01;INS;PRM;ANN
[1]  INS←INS01
[2]  PRM←CONFORM PRM21
[3]  ANN←ANN02
[4]  RES←1000×INS-PRM×ANN
      ▽

```

The *RESxx* function defines which insurance, valuation premium and annuity formulae to use for the particular plan type in question. The *CONFORM* function extends or replicates the “at issue” valuation premiums to a shape conformable with the annuity.

And now, for the actual formula programs:

```

      ▽ INS←INS01
[1]  INS←0.5×(M[XT]÷D[XT]) + M[XT+1]÷D[XT+1]
      ▽

      ▽ PRM←PRM21;NXN
[1]  NXN←N[X+PP]
[2]  PRM←M[X]÷N[X]-NXN
      ▽

      ▽ ANN←ANN02;NXN
[1]  NXN←N[X+PP]
[2]  ANN←0.5×((N[XT+1]-NXN)÷D[XT])+(N[XT+1]-NXN)÷D[XT+1]
      ▽

```

Even an actuary with no APL experience would recognize these formulae (programs) as that of a limited-pay life plan. The fact that we can almost see standard actuarial notation makes them easy to understand, modify and maintain. Note that the example is equally valid for Life to 65 as for 20 Pay Life; the distinction is made by the definition of the variable *PP* in the function *REPORT*, using fields 8 and 9 of the control matrix. In fact, the function *ANN02* is used for Level Term and Endowment plans as well as for limited-pay life plans. Also, no changes are required for the functions to work with select or ultimate assumption sets, due to the definition of the variables *X* and *XT* in the function *REPORT*. The overall flexibility of this approach makes it a powerful tool.

Conclusions

Any valuation system requires an investment in up-front thought, but if you organize with a view to running in APL, then the organization time is reduced. Furthermore, the time spent in preliminary organization is amply repaid in reduced programming time and run costs.

We have seen the advantages of defining packaged calculation environments, breaking down a prospective reserve formula to its simplest elements, and programming that formula to calculate any age and duration combination. We have full flexibility in our combinations of assumptions and formulae, unlike many other approaches which bind them together. This freedom to mix and match allows the actuary to adapt quickly and extend his formulae and assumptions, as and when the need arises.

We used a commutation functions approach, thereby retaining a high degree of readability through a strong semblance to standard actuarial notation. This can only enhance system maintainability in the years ahead.

Adapt these techniques to your own situation and define the assumption sets and formulae required for your purposes. I believe you will find the investment made in this approach well worthwhile.

APL IN RISK ANALYSIS

Ray Lemberg
President
Lemberg Consultants (Canada) Inc.
Montreal, Quebec

Lemberg Consultants (Canada) Inc. is an independent consulting firm specializing in risk analysis and risk management for capital projects and in special studies involving risk.

Our scope of services covers the identification and evaluation of the risks of:

- cost overruns
- delayed completion
- plant unreliability
- plant under-performance
- scope changes
- deviations from the project plan (i.e., contingencies)

We also formulate countermeasures for managing risks:

- contingency plans
- risk reduction through contract terms and conditions
- bidding and negotiating strategies
- real-time project control systems

In our work, we require access to data bases, special routines to analyze data, simulation programs (usually of a one-shot nature), and graphics output. We also have to be able to manipulate data, such as re-organizing a cost estimate into a different breakdown structure.

We are not and do not want to become programmers. For us, the computer is a valuable, labour-saving device which frees us to apply our efforts to what we know best: our risk analysis expertise.

OUR INTRODUCTION TO APL

We were introduced to APL about five years ago by the time-sharing bureau we were using at the time. A so-called APL freak questioned why we were using FORTRAN and special data base languages when we could do it all in APL. It did not take much to convince us that APL had very distinct advantages for us:

- APL does not contain the overhead of other languages (example: variables do not have to be DIMENSIONED as in FORTRAN).
- APL is smarter: it seems to know what it is doing (example: it knows the shape of variables).
- APL is geared toward modularity.
- APL is an interpreter so that errors can be fixed without starting again from the top.
- APL offers on-line interactive capability to manipulate data without having to write a special program.

From this first introduction, it did not take us long to convert. Unfortunately, the service bureau we were using was offering many languages and packages and we found that we could not operate exclusively in APL. However, the APL freak who introduced us to APL moved to I.P. Sharp Associates at about this time and proceeded to acquaint us with the I.P. Sharp system and its exclusive use of APL. We were convinced and switched over.

We have been very satisfied with the I.P. Sharp system and its capabilities. Furthermore, we have received outstanding assistance from I.P. Sharp in developing the special tools needed for our risk analysis activities. Because the tools are usually for a single application only, and the time to develop them is also very short, we have found that APL has reduced the programming time from weeks to a matter of hours.

EXAMPLES OF OUR USE OF APL

To illustrate the use we have made of APL and I.P. Sharp's packages, I will describe a few applications in risk analysis.

Cost overrun risk

We have, with I.P. Sharp's assistance in programming, developed a proprietary integrated system to assess the risk of cost overruns on capital projects.

We are able to determine:

- The probabilities of exceeding various levels of unescalated cost.
- The risk imposed by uncertainties in future escalation rates and the interest rates used for calculating interest during construction.
- The risk imposed by contingencies, that is, events which may or may not occur but, if they do, may impact the cost and schedule of the project.
- The financial risks in the payment terms and contract terms and conditions (used mainly for overseas projects where our client is bidding as a turnkey contractor) which affect cash flow, working capital, and financial exposure due to bank guarantees.

The system was developed over a period of time, starting at a very simple level. APL's modularity encouraged us to use this approach, adding features and functions as required by new projects and our advancement of risk analysis techniques.

The *COSTRISK* system, as we call it, started as a simple simulation routine to add together several cost variables, each variable having a specified probability distribution.

The routine was similar to the many Monte Carlo cost simulation programs on the market today.

We soon realized that the simple simulation had several weaknesses:

- It did not incorporate correlations (dependencies) among the random variables.
- It operated at too high a level of aggregation (e.g., a cost can equal units times the unit price and these two are the random variables, not the cost).
- It could not handle escalation or interest during construction properly.
- It had no effective way to quantify the effect of delays.

Recognizing the deficiencies, we set out to improve *COSTRISK*. A complete specification was prepared for a comprehensive system having all the capabilities we could foresee. We organized it into modules, the functions of APL. We decided to store data on file to conserve workspace size, and enlisted the expertise of I.P. Sharp to write the code.

The *COSTRISK* system is divided into four basic parts, each part having the appropriate functions to serve the controlling function. The controlling functions are described briefly below.

1. *SETUP*

This is a function which prompts for all the basic data needed for a project. It provides instructions if the user so requests. It asks for the descriptions of the probability distribution of each uncertainty factor multiplying the basic estimated cost of each cost item, and depending on how the data is input, it recognizes the form of the distribution as being either a uniform or triangular distribution. The interdependencies among the factors are also specified for use during the simulation.

After obtaining the data about the cost estimate in base dollars and its uncertainty factors, *SETUP* asks for a *SPREADMATRIX*, the rows containing the expenditure pattern for each cost item. The base dollar date, the start date of expenditures, and the date of the last expenditure are input. The frequency of the expenditures may be input as monthly, quarterly, semi-annually, or annually. Then the percent expended per time period is input for each time period, creating the *SPREADMATRIX*. A header is associated with the matrix indicating the date of each column of the matrix.

Next, the probability distribution of the escalation rate for each cost item is entered. The system recognizes either a uniform or a triangular distribution from the data entered. Dependencies among escalation rates are included.

Similar to escalation rates, the interest rate during construction is entered as a probability distribution. *SETUP* asks for the in-service date, the date up to which the interest is calculated.

SETUP computes *EPOWER*, a vector of exponents to be used in the escalation calculations. The elements are the number of years, in fractional form, from the base dollar date to the dates in the *SPREADMATRIX* header. Similarly, *SETUP* computes *IPOWER*, a vector of exponents containing the fractional number of years from the in-service date to the dates in the *SPREADMATRIX* header. All the data is placed on file.

2. DELAY

This function takes the *SPREADMATRIX* entered in *SETUP* and offers three ways to create new versions which represent the impact of a delay in the project.

Type 1 delay is simply a delay of the start of the project. The number of months of delay is input and the dates in the *SPREADMATRIX* header are adjusted accordingly. The new version of *SPREADMATRIX*, and its associated new *EPOWER* and *IPOWER* vectors are stored on file. If the delay is greater than the time from the new last-expenditure date to the original in-service date, the latter is moved to the month following the new last-expenditure date.

Type 2 delay allows the project to be delayed for a number of months during the project. The direct costs do not increase in this type of delay. The *SPREADMATRIX* header dates are altered so that the dates after the delay are increased by the duration of the delay. The in-service date is adjusted, if necessary, as in the Type 1 delay.

Type 3 delay is similar to Type 2 except that the direct costs may increase. New columns are created in *SPREADMATRIX* to cover the delay period and *DELAY* prompts for values to be entered into these columns.

Each new version of *SPREADMATRIX* created by *DELAY* is stored on file together with its new *EPOWER* and *IPOWER* vectors. By repeatedly using *DELAY*, one can create new versions of *SPREADMATRIX* from any existing version, thereby forming the effects of combinations of delays. They are all stored on file.

With APL, we are also able to directly access and alter a particular *SPREADMATRIX* version, if it is simpler to do it this way than to use *DELAY*. For example, often the labour costs may be delayed but procurement of materials and equipment is not. *DELAY* cannot distinguish between them so we have to intervene directly to shift the delayed equipment and materials row elements back to their original time periods. This is a simple matter using the rotate function of APL.

3. SIMULATE

This function performs the simulation and places the results vectors on file. It asks for the data to be used in the simulation. One can use combinations of a fixed value or the probability distribution for the uncertainty factors, escalation rates, or interest rate. The last two may be omitted entirely if desired.

For efficiency, one may choose that all of the *SPREADMATRIX* versions be used rather than just one. This way, the results vector for each *SPREADMATRIX* will contain values which are properly correlated with results vectors for the other matrices. One can then take differences among the results vectors to obtain probability distributions of differences of costs between various delay cases.

After asking for the number of repetitions for the simulation, the function proceeds to perform the simulation and place the results vectors on file. The vectors will contain one element for each repetition.

4. OUTPUT

This function provides a menu of the results vectors on file with a description that was input with each *SPREADMATRIX* created using *SETUP* or *DELAY*. From the menu, one may plot the probability distribution of a single results vector or the difference between any pair. Alternatively, one may obtain a listing of the statistics of each vector or the differences between selected pairs.

The *COSTRISK* system has become a state-of-the-art method of computing cost overrun risks, according to the comparisons we have made with other systems. No other system that we are aware of has the capability to handle escalation, interest during construction, delays and interdependencies. We are continually revising and adding to *COSTRISK* to make it suit the needs of a particular project. Needless to say, APL simplifies the task of making changes or adding functions.

Recently, we added a capability to handle multiple in-service dates (such as for a multi-unit power generating station where capitalization of a percentage of the total cost occurs at the in-service date of each unit). We have also added a net present value function which uses the reference data in the files of alternate projects to calculate the net present value of the capital expenditures for each project.

Because we are often involved in pipeline projects, we are devising a schedule risk system for such projects. The current CPM/PERT techniques are unsuitable for pipelines and other forms of linear construction such as railways, roads, transmission lines, etc.

Pipeline construction scheduling

The traditional approach to determining the time it takes to lay a distance of pipeline is to establish the daily rate from experience based on the pipe diameter and divide the distance by the daily rate.

A construction spread, the moving workforce laying pipe, consists of several crews, such as:

- clearing and grading
- pipe stringing
- drilling and blasting
- ditching
- wrapping
- laying in
- testing

We are developing an alternate approach, using APL-based graphics systems, based on drawing the time/distance performance of each crew. There are constraints on the distance separating crews, such as the maximum length of open ditch permitted, and the distance that the drilling and blasting crew must be ahead of pipe stringing.

The slope of each crew's line on the time/distance chart represents the crew's rate of production. Respecting the separation constraints, we plan to introduce the uncertainties in the crew production rates and determine the overall time through a simulation based on randomly selecting production rates from their ranges.

Using this approach, we will be able to determine the probabilities of various overall times to complete the desired distance of pipeline.

The production rates may vary due to several causes:

- weather,
- normal productivity variations, and
- unknown subsurface conditions, etc.

The pipeline construction model will enable us to assess the schedule risk in a better way than previously. The programming efficiency of APL and its graphics capabilities simplify the task of creating the model.

Iceberg scour risk

We were asked to assess the probability of an iceberg's scouring (and damaging) a proposed underwater power cable across the Strait of Belle Isle from Labrador to Newfoundland, a distance of 11 miles.

The engineering consultants, in their usual approach to risk, had recommended that the cable be buried in a trench, thereby avoiding completely the iceberg scour threat. What they failed to say was that the problem of creating a trench in solid rock in depths exceeding 70 meters posed its own risks. Trenching machines are not reliable, have not performed to such depths and require control from a surface ship which has to remain stable in rough waters to be able to guide the trencher accurately. Of course, trenching also costs more than laying the cable on the bottom of the Strait.

The client decided to have an independent appraisal of the iceberg scour risk. He wanted to decide whether the threat will be great enough to warrant the extra expense and risks of trenching.

In our analysis of the iceberg scour risk, we used APL for several aspects.

We analyzed the available data of the measured dimensions of icebergs in the population upstream of the entrance to the Strait. We obtained a histogram of iceberg drafts in 5 meter intervals. In addition, we were able to determine the characteristic waterline dimension for each draft class.

We used the digitized data of the bathymetry of the Strait to locate the channels along which icebergs of a given draft class may drift from the entrance to the Strait to the cable crossing location downstream. A "sill" of 70 meters maximum depth was located between the entrance and the cable crossing. It limits the maximum draft of icebergs that can pass through.

We used a specially-written simulation program to determine the conditional probability of scour of icebergs crossing the cable via various channels. The simulation took into consideration the possibility of icebergs reversing direction and scouring on the return passage over the crossing. About 10,000 icebergs were simulated to approach and pass through each channel at each draft class. The simulation determined the conditional scour probability per iceberg because it assumed that the icebergs would reach the crossing irrespective of whether or not they could actually get past the sill and survive the transit to the crossing location.

We used the array-handling capability of APL to set up the final calculation of scour probability. The number of rows were equal to the draft classes from 0-5 to >70 meters. The columns contained various data for each draft class:

- (a) The frequency of occurrence of icebergs.
- (b) The "presentation" width for icebergs upstream of the sill. This is the transverse distance over which icebergs in a given draft class may be distributed upstream of the sill.
- (c) The transverse width of the opening in the sill.
- (d) The conditional probability of scour at the left and right edges of each channel at the crossing (determined from the simulation).

The annual probability of scour per iceberg entering the Strait per year was equal to $(a) \times [(c) \div (b)] \times (d)$ above. The average annual number of icebergs entering the Strait was estimated by taking the difference between the number of icebergs north of and south of the entrance to the Strait. Those which do not enter the Strait travel around Newfoundland. APL provided us with programming efficiency—none of the functions took longer than a few hours to write, test, and implement.

Storage tank capacity

As part of a reliability study for a client, we determined the peak LNG (Liquefied Natural Gas) storage capacity required by a vaporization terminal. This sub-problem was interesting in that the terminal was designed to operate at a continuous vaporization rate and a 20% reserve to overcome downtime. Shipments of LNG to the terminal were to be at irregular intervals from the Arctic. The results of this study will be considered in the contract negotiations between the terminal operator and the LNG shipper.

We devised a simulation program in APL to consider a nominal 33 shipments of LNG per year. The load per shipment, and the interval between arrivals of shipments were considered to be uniform random variables within a specified range. For a particular combination of load and interval ranges, we determined randomly selected loads and intervals. Keeping the withdrawal rate from the tanks and unloading rate of the ships constant, we were able to plot (in memory) the fluctuations in the LNG tank level as a function of time. The peak level was remembered as well as the total time for 33 shipments, and the number of days per year the tanks were empty. The total LNG delivered times 365 divided by the time for 33 shipments gave the annual throughput.

The number of tank-empty days in the total time provided the non-utilization factor for the terminal. The simulation was repeated 300 times to produce a 3 by 300 matrix of results.

From the 3x300 results matrix, we sorted each row and thus each column represented a percentile. We selected the 50th, 75th and 95th percentile results for tank level, annual throughput and terminal (plant) utilization.

The results were obtained for shipment intervals ranging from 10-12 days to 7-20 days. The plant was designed for an average of 11 days. The results showed that for a 7-16 day interval (the proposed shipping schedule), the LNG storage capacity would have to be doubled in order to be able to operate at a 75% level of confidence of being able to unload each shipment. Furthermore, the annual throughput would average about 80% of planned throughput and plant utilization would be about 90%.

The results pointed out the need for the terminal operator to:

- Fix a shipment interval in the contract,
- Not guarantee the annual throughput,
- Establish a penalty rate for tank-empty days,
- Be allowed to refuse to unload a ship until the minimum interval had passed, and
- Accept liability for demurrage charges if unloading took place after the maximum interval had elapsed.

Because the terminal had an equivalent storage facility in the Arctic, operated by the shipper, it became apparent to the shipper that he also would have to double his storage capacity.

Again, the programming time in APL was on the order of a few hours. The results were obtained quickly, at low cost, and in graphic Technicolor (thanks to SUPER-PLOT and EASYGRAPH).

ACKNOWLEDGEMENTS

We are grateful for the assistance and enthusiasm received from I.P. Sharp Associates. To Paul Hansuld (Toronto), our thanks for his diligence in preparing the bulk of *COSTRISK*. Tony North (Montreal), has continued to provide cheerful and expert assistance with our various needs for special routines, plotting, data base access, and revisions to *COSTRISK*.

ADAPTING A FINANCIAL PLANNING SYSTEM

Robert A. Besse
Manager, Decision Support Systems Section
Operations Research Department
McDermott Incorporated
Alliance, Ohio

INTRODUCTION

During the last 6 years at McDermott Incorporated, the Operations Research (OR) department has developed in APL a number of financial planning systems. These systems are used by financial analysts and managers at most McDermott locations in North America, Singapore, Dubai and several western European locations. These systems provide a variety of services to division and corporate staff including:

- 1) Basic preparation and transmittal of financial forecasts
- 2) Ad hoc querying, analysis and reporting of financial forecasts
- 3) Terminal and pen plotter graphics
- 4) Financial forecasting data bases which may be accessed by models designed to address senior executive concerns about the business

This paper first discusses McDermott Incorporated, and then recounts the history of APL usage at McDermott as the APL financial systems have grown to become the backbone of monthly, quarterly and annual financial forecasting activities involving several hundred people around the world. The rest of the paper is a summary of the major changes required in our systems and methods as the financial systems were adapted and augmented to address an ever increasing management forecasting and decision making requirement.

BACKGROUND

McDermott Incorporated is a comprehensive energy services company. Our operations are organized into two multinational units—McDermott Marine Construction and Babcock and Wilcox.

McDermott Marine Construction designs, builds and installs offshore platforms and marine pipelines used in drilling for, producing, processing, and transporting oil and gas. To support its worldwide operations, Marine Construction has the largest fleet

of marine construction vessels in the world. We have fabricated and installed more fixed platforms, in depths from ten to more than one thousand feet, than any other company.

Babcock and Wilcox engineers, manufactures and erects complete fossil-fueled and nuclear steam systems. Fossil-fueled boilers are used for power generation and industrial processes. We also manufacture high quality specialty steel tubular products for applications such as deep, high pressure oil and gas wells and power generation. Additionally, we manufacture specially engineered insulating and refractory products for high temperature industrial applications.

For our fiscal year ending March 31, 1982, McDermott Incorporated realized net income of \$213 million on revenues of \$4.8 billion, and we employed 59,000 employees.

APL was first used in 1974 by the OR department which was then a corporate function of Babcock and Wilcox. During these first few years, APL was used experimentally to evaluate its appropriateness as a modeling language. The expense of using APL on commercial timesharing combined with a lack of experience with interactive models and systems limited the use of APL to a few fairly simple applications. Early in 1977, the first financial planning system designed to collect and consolidate financial forecasts was developed. APL was chosen because it was interactive, and could be accessed on timesharing from Babcock and Wilcox locations by phone. Also influential was our timesharing vendor's claim that APL was 10 times more productive than other languages for this kind of work. Little did we anticipate that we were just beginning a long but rewarding education in issues like human engineering of interactive systems, design of large software systems, design of data base systems, user documentation and education, and coordination of users around the world.

Four months later we had an interactive planning system implemented in all major Babcock and Wilcox locations. This system allowed planners to prepare various alternative 5-year forecasts of earnings and cash flow by major product groupings. These forecasts could be consolidated and reviewed by group and corporate management. Although this effort was quite exhausting for the people involved, the success of the system in such a short time sold us on APL's productivity.

In 1978, Babcock and Wilcox was acquired by McDermott and the OR department became a McDermott corporate function reporting to the Director (now Vice President) of Corporate Management Information Services. Whereas Babcock and Wilcox had been interested in long-term financial forecasting, McDermott interests were directed towards forecasting income statement and balance sheet items each quarter going out for the next 4 quarters. There was also increased interest in forecasting cash-flow and capital expenditures. Early in 1979 two more systems were implemented in Babcock and Wilcox which supported the quarterly forecast and capital expenditure forecasting activities. Both of these systems took only a few months to develop even though a major reorganization of the financial analysis group occurred during this period.

The success of the Babcock and Wilcox systems led to their being adapted to each of the Marine Construction units, an activity which was completed in 1980. Meanwhile other related APL financial systems and models were being developed by OR for the Treasury, Corporate Planning and individual divisions, each of which needed specialized, sometimes quite complex models to study special problems in investment, marketing and detailed product line forecasting.

By this time the APL usage on commercial timesharing had grown to over 100 APL accounts.

A project to migrate the workload onto McDermott's own computer began in August 1980 and was completed at the end of March 1981. We presently run SHARP APL on an IBM 3081 using a variety of in-house and commercial telecommunications arrangements. The APL workload doubled during the first year of operation since coming in-house. Most of this work relates to financial planning and reporting systems. By mid-1982, over 300 APL accounts were in active use on the system with over 40 sessions running simultaneously at peak periods. The number of APL'ers in OR currently stands at 16. With the success of these systems and the high degree of favorable financial user reactions to them, the growth of APL at McDermott will continue for some time. Although OR now has two full-time employees located at the New Orleans corporate headquarters, the promotion, design, development, implementation, and continued worldwide user support of these APL systems has been and continues to be directed by the OR personnel located in Alliance, Ohio.

Without APL, this story would never have happened. But using APL did not automatically bring success. We learned to manage APL and the rapid growth and change in systems that APL can create. We learned to make APL work for us and we changed our ways of doing things as we adapted our initial single financial reporting system into a world-wide network of interrelated financial planning systems and data bases. The remainder of this paper discusses the major ways we have changed systems and practices during the last few years.

BEING NICE TO SOMEONE

From the time of the first major system, the philosophy behind the design was that financial analysts and managers did not want to be programmers, especially APL programmers. Instead, we felt that this user community needed useful, adaptable computer support to aid them in the routine calculations and reporting activities which consume much of a financial analyst's time. If numbers could be quickly entered and changed and reports quickly produced, then more time could be spent reviewing the quality of the forecasts and less time spent performing the mechanics. At the same time, individual financial analysts had different computational needs; some user control had to be exercised over when data was to be changed, when computations were to be performed and when reports were to be printed. On top of these considerations were our own expectations that some of the intended users would not easily accept the concept of a centralized computer program for reasons like: data confidentiality ("Who's going to see my forecast?"); ease of use ("I can do it faster on my calculator."); and lack of faith ("This kind of thing can't be made to work!").

Several of the initial design features worked well and are found throughout our APL systems today:

- 1) Strict avoidance of any "computerish" or "APLish" symbols or jargon. There were to be no deltas, quads, "system aborted" or "Abended" messages to confuse users.
- 2) Use of command modes whereby a prompt like *REPORT OP?* would inform the user that he could now type any number of English-word commands optionally followed by a list of English-word parameters and/or data. Useful sequences of commands may be entered by the users at their discretion to change inputs and display results relevant to the task at hand.
- 3) Destruction proofing using APL exception handling to eliminate damage

should the user depress the break key, and to protect the session and reduce user anxiety should the user encounter a program error in the system.

- 4) Design of inputs and reports so that they look equally nice on APL terminals as well as on non-APL terminals and line printers. This allowed divisions to run the systems with existing equipment.

We learned many things along the way by listening to comments from our users:

- 1) Our on-line documentation or *HELPS* weren't always helpful. Many were not well phrased, some were far too long and several were incorrect. As the systems changed, the *HELPS* didn't and became obsolete. APL came to our rescue. We developed a modular text editor which could be called when running the system. This editor allows a privileged non-programmer user to print and alter *HELPS* while running the system, thus increasing the chances that the *HELPS* are kept up to date.
- 2) Common user input and command errors are checked for and messages are displayed explaining the error. We discovered, however, that many users got worried when they entered a proper command yet the computer only came back with another prompt. We now print confirmations of successfully completed commands as well as error messages for the unsuccessful ones.
- 3) As the number of systems increased, we encountered another user difficulty. A growing number of our users were using two or more of our systems. They asked questions like: "Why do I say *LIST* in this system but must say *PRINT* in that system to do the same thing?" or "Why does this command do one thing in this system and something very different in that one?" This is a difficult issue because there are legitimate differences in the workings of similar commands in different systems. We now formally review the choices of command names and other keywords in an effort to guarantee as much consistency as possible.

Recently we put several very popular features in all our systems. Users may abbreviate keywords to reduce typing. They may also enter several commands at once, separated by semicolons, if they are sure of what they are doing. Finally, they may predefine and save text strings of commands or other inputs, which may be invoked at any prompt by typing a "*" followed by the name of the word. We call these strings "starwords". Starwords may be created and edited during a session and may contain references to other starwords. Some users have become adept at defining their own starwords to produce their own reports or to greatly simplify routine use of a particular system.

SPREADING THE WORD

Training the users to run our systems has grown to be as much a preoccupation of the APL development group as developing the systems themselves. Initially, our user instructions were brief and not always effective. On the first system, we sent all of our analysts at once out to the field for on-site training in a very hectic short period of time. We discovered that there was some job changing going on in the divisions as users we had trained were often reassigned other duties. Explanations given to users often were not passed on to their successors. The phones in OR soon started ringing off the hook and a few people began wishing they had unlisted numbers. Although

the responsibility rested squarely with the users to train their own people, they sometimes had little time to do the job.

Several fundamental changes improved user training:

- 1) College students from a nearby university were hired as summer and part-time employees to spend more time writing user instructions.
- 2) A full-time (non-programmer) person was hired to coordinate the writing, editing, typing, publishing and distribution of user instructions.
- 3) Word processing equipment was acquired and used for all user instructions, reducing the effort to produce frequent updates as systems grew in capabilities.
- 4) Commands in systems were made more consistent, allowing the use of the same explanations in multiple user instructions. This change allowed more time to be spent describing individual commands.
- 5) Outside experts were asked to review and make improvements on the style of documentation produced.
- 6) A news service was added to several of our systems using modular APL routines similar to the ones used for the on-line *HELPS*. When loading a system, users are given headlines about recent changes to the system they're using. Once they print an item, they no longer are bothered by the headline for that item.
- 7) Corporate users have been enlisted to provide day-to-day assistance in fielding questions from users. Not only has this change reduced the phone calls to the APL developers, it has also increased the dialogue between division and corporate financial staffs on operational issues not related to the APL systems. By and large these corporate users, which we call corporate system "stewards", want to spend time on this activity and do a good job at it.
- 8) Training sessions are now held at central locations for personnel from several divisions. This change reduces efforts by OR personnel and corporate staff. It also lets division users meet each other and share ideas on using the APL systems. Finally, these group training sessions afford the corporate staff an opportunity to discuss other issues with the division people.
- 9) User operating problems in overseas and other remote locations can now be monitored. Diagnosing someone's problems by phone is often time consuming and ineffective. By using the S-TASK and shared variable capability of SHARP APL, we can now have a user in Tokyo or Brussels turn on a monitor while he runs his system. This allows a person halfway around the world to simultaneously see what the remote user is entering and printing on his terminal. Monitoring, which can only be turned on and off by the person being monitored, has enabled user problems to be quickly diagnosed. The "tutor" may even reverse the process, monitor himself, and have the user in the field watch how the tutor runs the system. In one instance, these techniques were used to train an overseas user in Tokyo.

GETTING THE JOB DONE

Our most profound changes occurred in the way we use APL. Initially, we had only a few people who knew APL, and their knowledge was only at a beginner's level. Most OR people used FORTRAN, SIMSCRIPT or other compiled languages. One by one, most of the people were exposed to APL, sometimes because it was the only way to do the job on time. Our user community was getting such fast, good results from our APL systems that they began asking for more and expecting faster turnaround. Our problem was to stay on top of our successes and keep a step ahead of our users' expectations, yet not let the quality of our support to the users degrade.

We eventually realized several factors about our environment:

- 1) We couldn't afford the 3 to 6 months waiting time it seems to take for someone to become proficient in APL.
- 2) Many of our applications were similar to others and in fact could be divided into 3 or 4 generic groupings.
- 3) Some of our people were really excited about APL and its potential for large-scale systems and generalized tools.
- 4) Most of our people were more excited about doing modeling and other applications quickly for our users and viewed APL as an excellent means to that end. They often did not want to know what dyadic transpose does or how to share variables unless they needed to use those things.
- 5) With the APL financial systems growing in use and importance, we were becoming inconvenienced by people going on vacation or every now and then leaving the department. Backup of OR people responsible for each system was needed.
- 6) The major proportion of APL programming time was spent formatting reports, designing input prompting schemes and working with APL files.

The only way to cope with the rapid work load growth was to develop tools that people can use quickly to implement different systems. The tools we now use are each large collections of APL functions along with documented programming conventions explaining how they are to be used to implement various sorts of systems. Each tool could be thought of as a computer language on its own. Our goal became one of trying to minimize the amount of original APL coding, yet retain the power of APL when needed.

Tools grew with each application. We learned to generalize our software to handle each new requirement. Sometimes, we had to do a major rewrite of a tool to incorporate a significant new capability and to undo problems of previously weak or cumbersome code. We also learned that conversion from older generations of tools to revised ones could be time-consuming, so we now support our applications people with more conversion routines to upgrade their systems. We also learned to include "hooks" or functions that do nothing but which are called at certain well-chosen places in the code. These hooks allow analysts to use custom logic for special things without interfering with the underlying code of the tools being used. Below is a brief discussion of the major software tools we have developed to meet our needs.

Financial modeling language

Many applications require defining names and logic for a number of rows and columns. Our modeling language is similar to most APL based modeling languages and has been very useful. We provide a set of calculator mode functions with English word names that define and redefine row and column names as well as default format control options. Files to save models as well as alternative case studies are also easily created and modified. Users may control report formatting from calculator mode as well as from simple APL functions. Users of the models may easily alter any input number to do quick sensitivity analysis. A typical calculation function might be:

```
▽ CALCULATE
[1] 1 GETS SUM 1.1 1.2 1.3
[2] 2 GETS (2.1 TIMES 80) PLUS 2.2 TIMES 81
[3] 5 GETS 1 DIVIDEBY 2 TIMES CONSTANT 2.187
[4] COLUMNWISE
[5] 1982 GETS SUM 82.1 82.2 82.3 82.4
▽
```

All numbers refer to line or column numbers unless preceded by the word *CONSTANT*. Each keyword in the above expressions is a function. We learned that if we used only our keywords (and not use complicated APL logic), we could do some fancy things. First, we could rewrite these keywords (i.e., *GETS*, *TIMES*, etc.) so that they would build up a character matrix of APL code which would be impossible to read but would run efficiently. In effect, we “compile” one form of APL function into another allowing us to easily maintain the “English” version, yet execute the “APL” version.

We discovered that another version of the *GETS* and *TIMES* type keywords enabled our original calculation functions to generate lengthy documents spelling out what each equation does. This output has helped our users understand the logic of the program.

New people have come into our department and produced a model during their first week. One model was written in which users can generate a custom capital project evaluation model by answering some questions regarding number of products and depreciation methods. Based upon their responses, lines as well as logic are automatically defined as a model is created for their own use.

Most of our models start small and seem to increase in size. Workspace becomes worrisome after a while. We recently modified our modeling language to handle a larger number of rows than can fit into the workspace at once. We are also developing risk analysis routines for the language.

Financial network reporting software

This software is similar to the modeling language in that it is oriented to row-and-column calculations and data storage. It differs in that it has become a full blown data base system designed for our major financial reporting applications. This software supports the user dialogue and extensive data base sharing and integrity logic. Non-programmer corporate stewards may exercise control over which divisions or product lines are installed in the system, how they are to be consolidated and which user accounts may view what data.

Corporate users are prohibited by the software from viewing divisions' data until it is officially "SENT" by each division. Divisions are prohibited from changing data when it is officially "FROZEN" by corporate stewards. Users may create custom reports which may be saved and easily run for many sets of forecasts at one time. Terminal and pen-plotter graphics are also provided for. The data checking and control logic has reduced inaccuracies in the financial reporting process as well as shortened the time required to compile a company-wide financial forecast.

Recently, there has grown a need to share data between systems. Under carefully worked out arrangements, messages containing data are now being sent by users in some systems to other users in the same or different systems. We've developed a programmer tool called DELTAPIPE which allows any arbitrary amount of data to be sent with descriptive information from one user to another. Our computer systems routinely send and look for datasets in a way similar to the way people use electronic mail systems. This protocol has greatly simplified the potential nightmare of grabbing data from one system and pushing it into another when updates to our data bases are done continuously by many people in different locations.

Record oriented data base language

We learned that many financial applications require routine and ad hoc reporting from traditional record-oriented files. Data bases of individual items such as investments, assets and contracts have been developed using this APL tool. The users can have a language which permits them to select and sort records using arbitrary criteria and then produce detailed and summarized information from selected fields. Our software is similar to a growing number of commercially available systems generally known as "fourth generation languages".

We've made many changes to this software, as our financial users have become aware of the ease of doing personalized ad hoc data analysis using its inquiry and reporting capabilities. Once a user sees what this software can do, he often starts to think of other applications.

This language provides a hierarchical data structure with one-to-many data relationships. In multi-user applications, users can be prohibited from seeing other's data. We also support 3270 forms inputting as well as highspeed laser printing output. We have had to address many of the typical problems that data base languages must address.

This language has its own data definition language. This means that a new APL'er can learn an English set of commands which permit him to write simple APL functions that will define the data base. We've had people set up new data bases in a morning, after spending a day learning the language.

Our language is by no means as complete as commercial languages but it does allow us to easily intermingle APL models with our data bases. Several of our financial reporting systems are also using this record-oriented data base software to collect transaction or other information related to the information stored in these row and column data base reporting systems. We also use DELTAPIPE to share information between these data bases and other systems. Shortly, we'll be able to take general non-APL files and make batch updates to our APL data bases. Like all other APL work our data base applications have grown considerably in size. We can now support 100,000 records per any one table. So far, this has been adequate for our financial applications.

Software development tools

All of the tools mentioned so far, as well as many of the applications we support, involve the equivalent of many workspaces to hold the programs. These systems have become too large to be easily maintained by only one person per system. We have found ways to allow several people, sometimes located in different places, to effectively work on the same programs at the same time. Using the SHARP APL package data type, we have organized our functions into groups on APL files. Functions required for any one program may come from different files. This scheme allows us to easily share code from our major tools. Everyone running our data base language is using functions from the same place. However, each application may employ its own groups of functions along with the standard ones. Custom functions are combined with standard functions allowing systems to work differently yet basically be the same program. Workspace management routines bring functions into the workspace as required and remove them when they are no longer needed. This automatic workspace management is straightforward to set up and hardly visible to the programmer while development is taking place. The overlaying logic may be easily altered under program control while the application is running. This allows us to have users running different variations of our systems at the same time.

Utilities allow us to locate and copy in functions from different systems. Functions must be individually saved, which takes getting used to, but we can now determine who last saved each function and when. We use APL workspace documentation and function updating tools. We've adapted these tools to work on our files of program functions.

We use a character text editor to great effect including the on-line *HELPS* and news modules mentioned earlier. The character editor also speeds up editing of large character matrices, latent expressions and complex trap expressions. We are working towards linking our on-line *HELP* system with the program maintenance routines so that system documentation may be saved at the same time program changes are saved.

Users load systems via a special program so that they only need to know the name of one workspace. This program prompts for the systems desired, records the request and prompts for a password if necessary. The proper system is then loaded if the user is permitted access. Certain user IDs for each system are allowed to change user enrollments or other access control information. This provides standardized system usage data and access security.

Analytical and other modeling tools

The most important direction we are now moving in is towards ad hoc computing in support of executive decision-making. The data base reporting systems we have implemented provide a wealth of current financial performance forecast and historical information. The management of the company is discovering the usefulness of this data resource and is asking questions which can be answered by using it. Through time, the questions that will be asked by top management will require more and more "soft" data, meaning estimates of missing information modeled using the "hard" data collected by the systems.

Many of the traditional Operations Research/Management Science modeling tools will have to be linked up with our financial data bases. Already we have interfaced our APL data bases and report writers with IBM's MPSX linear programming product. We have developed pieces of an econometric modeling language. It is conceivable that

we will need to interface simulation languages with APL as well as network modeling and decision analysis systems.

The goal of these linkups will be to take advantage of the power of these traditional modeling tools yet receive the productivity, flexibility and ease of user access typical of our APL systems. Combining the MS/OR technology with the APL interactive data base/analysis/reporting technology should permit us to respond quickly with new insights relating to key concerns of senior management.

GETTING THEIR ATTENTION

Perhaps the most interesting development with our APL financial systems has been our success in involving financial analysts and management more directly in the system design process. We believe APL is a good prototyping tool. We can model our requirements and get results quickly. It is often suggested that after a prototype is built the system be rewritten, in some more "efficient" language. The fact is, our systems are continuously evolving and we have never wanted to freeze their specifications so we could convert the APL to something else. These financial systems keep changing because the environment concerning our key financial and strategic questions keeps changing. Therefore I like to call our APL systems "self-prototyping". We can show our users the impact of a proposed change. This ability has made the dialogue with the users more effective and has resulted with many new ideas. Our users have more appreciation of the many considerations that must be weighed against each other as a system is designed. Their requests have become more reasonable through time as they can watch a system develop.

Our data definition language and our modeling language are proving to be effective tools at involving non-programmer users in potentially complex issues of data base design and model logic. To use the data resource effectively, the users must manage the data and its use. They must have a good understanding of what is in the data base and how it got there. If the APL systems are mere "black boxes" to them they won't realize the potential that is there. The problem is that most people won't be good APL'ers and shouldn't waste time trying. But also many APL'ers won't be good at advanced issues of data base and model design. The solution is to have the expert APL'ers do a better job at explaining the issues and choices to the non-expert users and soliciting their input. Our APL tools are proving to be an excellent catalyst in fostering this vital working relationship between the APL modeler and financial management.

CONCLUSION

APL is a powerful tool and it can be effectively used to develop large financial forecasting and other decision support data bases and systems. One should be aware that although APL permits the job to be done more rapidly, this does not mean that the job is trivial. Our OR group made a major professional commitment to providing McDermott with decision support systems. Our management has made a major commitment to managing a growing number of these systems. Consequently, much has happened in a short span of years.

The momentum that we now have permits us to build upon this basic information foundation which senior management needs to make the quick, tough decisions which today's marketplace demands. Increased use of models and graphics will continue to

evolve as we continue to discover how to make our systems even more responsive to the executives' needs. The effort made has and will continue to pay good dividends to the company!

ABS - A BUDGET SYSTEM

Maurice Elliott
Coordinator, Financial and Operations Planning Systems
Dome Petroleum Limited
Calgary, Alberta

Introduction

Until 1979, Dome used a batch computer system for the preparation of its annual budgets. This system gave users some flexibility in the calculations and reports. But as the volume of data grew it became more and more an unwieldy tool, largely because of its batch (punched-card) orientation. With the tremendous growth of Dome since 1978, the Budget Manager looked for a more usable system. Many existing packages were examined. Among them they contained all the facilities the budget staff had identified as requirements. But no single package offered enough of them to warrant serious consideration, so the decision was made to write an in-house system tailored to Dome's needs.

At the same time, Dome was making increased use of the I.P. Sharp system, particularly for its economic evaluations of oil and gas ventures (using the GOFER package). Usage of GOFER was expected to grow as Dome expanded, and this, coupled with the feeling that APL would be an ideal language in which to develop Dome's planning systems, led Dome to justify running SHARP APL on its own computer. Consequently, the SHARP APL system was installed at Dome in March 1980, and became available to users two months later.

Coupled with this, Dome increased the number of systems staff devoted to planning systems from one to three in May 1980. I was one of the additions, and was given the task of developing the system to support the preparation of the budgets.

The remainder of this paper describes the development of this system and its overall architecture. I also give some examples of the tasks for which it has been used by business analysts (who, in general, have no programming experience).

Establishing the principles

When I joined Dome, I had had no prior experience in any of the businesses in which Dome is involved. My first task, therefore, was to gain some basic understanding of what Dome was trying to budget for. I was immediately thrust into a series of meetings where people from various departments explained how their part of the company operated, and how they went about preparing their budgets.

Dome's business has many facets. The best known is its exploration program in the Beaufort Sea. But Dome also has an active oil and gas exploration and production program on the Canadian mainland, as well as interests in other countries. Dome also operates a number of natural gas liquids processing plants and transportation networks (i.e., pipelines). The budget system had to satisfy the needs of all these areas, as well as those of the purely administrative departments.

The work of preparing the annual budget at Dome starts in August. Thus I had a bare three months to get enough of the system up and running in time to be useful. It quickly became obvious that I had no hope of grasping all the details of each department's requirements in time to program them. Furthermore, even if I could, the resulting system would rapidly become obsolete in the dynamic environment at Dome. We would have to dedicate too much systems resource to maintenance of such a system, even though it would satisfy only the needs of the budget process.

Looking around the company, we realized there were many examples of business analysts using their calculators for financial analyses where it would be quicker and more effective for them to use the computer. While raw APL is an excellent tool for this kind of work, we did not feel that most of the business analysts would have the time, or the inclination, to learn it. What they needed was a higher-level language that would relieve them of the many system organizational problems while performing the calculations they specified.

As a result of this, I felt it would be worthwhile to parameterize the budget system to the extent that it could satisfy the needs not only of the budgeting process, but also of these financial analyses. Moreover, I could delegate the parameter-setting process itself to the users, and relieve them of the need to work through a third party to have their needs programmed. The Budget Manager (an ex-systems analyst) quickly saw the benefits, and pitfalls, of this approach. He fully supported our recommendations (provided his budget was not jeopardized, of course).

We held a design review at which I proposed to develop an open, rather than a closed, system. This was justified on the basis that:

1. I would not need to absorb so many details of the company's operations.
2. The users could specify their requirements directly to the system itself, rather than through an intermediary.
3. Users would easily be able to respecify their calculations or reports as the need arose.
4. The system would be useful for many more tasks than just the budget—thus its value to Dome would be much greater.
5. To cap it all, I claimed I could have an open system up and running as quickly as I could the closed one, because I would defer the need to learn so much about Dome.

On the other side of the coin, we realized there were disadvantages to this approach:

1. The system would be more complex on the technical side, because it deals with the problems more abstractly. Thus, maintenance would be more difficult though we expected there would be less of it to do.
2. The system would be more costly to use, because it would create less-efficient code than a good programmer could.
3. By putting such a system in the hands of users, we were exposing the machine to the possibility of wasteful usage, since users often do not recognize the impact their requests have on the machine.

Following this design review, I proceeded with the implementation of the open system I had proposed.

Design considerations

An open system, such as this one, tends to intimidate users more than a closed one. Many of the prompts it prints have to be worded more generally. And the user often has a far wider variety of replies from which to choose. Some users never seem to grasp the extent to which they are in control of the system. And some fail to comprehend how to connect the options the system presents to them in order to solve their particular problem.

Because of these difficulties, the success of an open system is crucially dependent upon the quality of its user interface. The system must use terms familiar to the user wherever possible, and must be consistent in its terminology. And the user must be able to ask for help if he is unsure what to do at any point of input. Yet the interface must not be so long-winded as to frustrate and bore an experienced user. He must be allowed to anticipate prompts and enter many replies on a single line of input, if he wishes. Many prompts have default replies, which appear in the prompt. To accept the default, the user simply presses the ENTER key. Many other common replies (*HELP*, *END*, *ABORT*, etc.) are reduced to a single character.

Another feature of the user interface in ABS is its use of a "compiler" to tailor messages to the situation at hand. Figure 1 shows how this technique is used to print error messages that are very specific to the problem, yet keep the code in the workspace from becoming cluttered with the details.

Figure 1

Example Of The Use Of Compiled Messages

APL Code in Workspace

```
BAD←~CODES∈VALID
→LABEL IF (0<ρBAD) ERROR 123

▽ R←R ERROR N
[1]  ⍎(0=⍋NC 'R')/'MSG N⍋→0'
[2]  →R+0 ⍋ MSG N
▽

▽ R←MSG N
[1]  R←SCOMPILE ⍋READ MFN,N
▽
```

Message 123

THE FOLLOWING CODES YOU ENTERED ARE NOT RECOGNIZED:

*{(PW,0) FOLD VECTOR DECODE BAD}
PLEASE RE-ENTER*

A second design consideration arose from the expectation that the system would be large (it now consists of about 1.5 MBytes of functions and variables). Obviously, the code would have to be paged. But also, since I am a firm believer in writing small functions, a large number of function names would be required (over 1500 at the last count). Therefore, I established at the start a number of function naming and structuring conventions which I applied strictly throughout the system. Knowing these conventions, you can easily find your way about in most areas of the system. Naturally, I also paid great attention to breaking the system up into functions which are small and highly cohesive, with the minimum of coupling between functions. These considerations are essential if a system as large as this is to be robust and maintainable.

The third design consideration—a major one in the users' eyes—was to strike an appropriate balance between the users' desires and what is implementable with reasonable efficiency. Users, particularly those at Dome, tend to ask for the moon as soon as you start promising them flexibility. A great deal of discussion and horse-trading went on, during which I was weighing the importance of each request against the difficulty of implementing it while retaining the generality of the system. In general, I resisted requests which would have served the needs of only one user, though most of them were subsumed by more-generally useful features of the system. On the other hand, I quickly agreed to install features that would have widespread applicability, even though some of them were not at all trivial to implement. It was this horse-trading that occupied the majority of the three-week discussion with users that preceded my presenting a proposed design for the system.

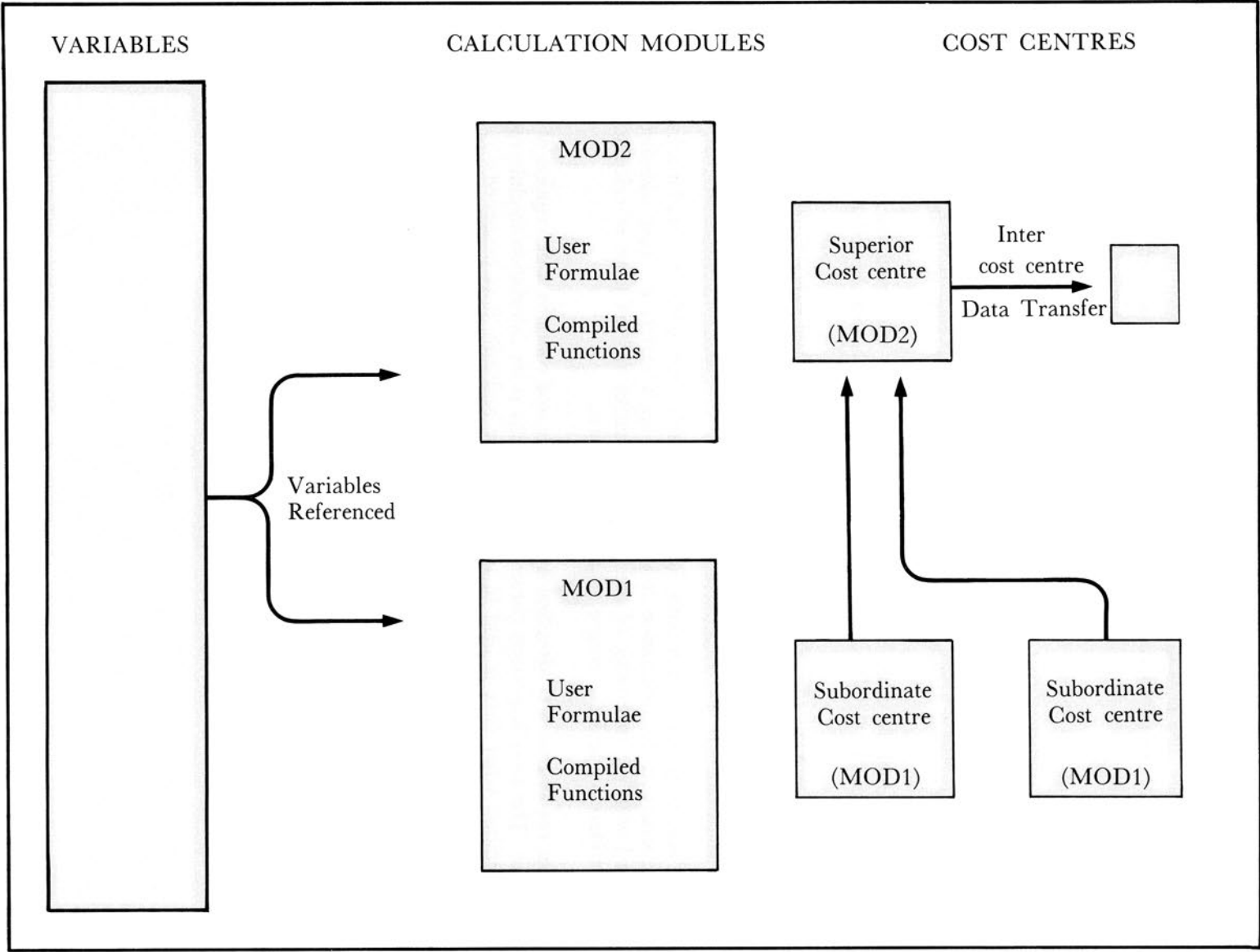
Major components of the system

After three weeks of cramming, I was asked to prepare for a walkthrough at which to present my proposed design. Naturally, at this point I could not go into great detail, but I did make it clear that the system would be very general, and described what the major phases would allow the user to do.

ABS users have the ability to create and destroy their own databases. The creator of a database may give access to other users. All parameters which the user supplies to ABS are local to the database in which they are defined. But an inter-database transfer phase allows users to copy all or parts of one database into another. Thus users can work with whatever degree of collaboration they wish. The overall structure of an ABS database is shown in Figure 2.

The first phase the user typically enters after creating a database is the variables maintenance phase. Here the user defines the variables which are relevant to the problem at hand. Each variable has a short code by which it is identified. It also has a description, format phrase, data type and (perhaps) a constant value. The format phrase may be just the number of decimals, or it may be a full-blown *FMT* format phrase. The data type tells the system two facts about data associated with this variable: (a) whether it is "income statement" or "balance sheet", and (b) whether it is "gross" or "net". The system's treatment of income statement and balance sheet data differs in some areas (e.g., the default calculation of the total column when the user has not supplied a specific formula). Net data is the share of gross data that is apportioned to the partners in a joint venture. The system automatically holds net data by partner, and performs the apportionment when requested. A variable's constant value is used by the system when no other value is supplied. It is useful for data that has the same value over many cost centres, such as the number of days in the period, world price of oil, conversion factors, etc. Except for the code itself, and the data type, all parameters of the variable can be overridden at various points in the system.

Once the user has defined the variables, he can specify the formulae that relate their values. Formulae are grouped into calculation modules. Each module specifies the calculations to be performed on the data in a cost centre. Each cost centre may only have one calculation module, though similar cost centres may share the same module. For each variable whose value is to be calculated, the user enters an APL-like formula expressed in terms of functions and other variables. A subset of APL functions is available, and the formulae follow APL syntactic and semantic rules (i.e., they are evaluated in right-to-left sequence). Most formulae are general, that is, they are applied to any periods where the system sees fit to apply them. The user may also enter a formula for a specific period, for example, to calculate a weighted average in the total column. If the user attempts to refer to a variable which is not defined, he is warned and may define it at this point, or the system will reject the formula. The system also rejects formulae at this point if their syntax is invalid. As the formulae are entered, the system remembers which gross and net variables have been mentioned. Then, when the user associates this module with a cost centre, the system deduces which variables' data needs to be accommodated in that cost centre.



Database Major Components

Figure 2

When the user has entered all his formulae, the system compiles and saves APL functions which will perform the required calculations on the data for any cost centre. One essential service which the system performs for the user at this point is to arrange the user's formulae into a computationally valid sequence. Thus the user can enter his formulae in any order, and need only ensure that all are present and correct. The system handles two special cases which are particularly helpful to the user. If the formula for a variable refers to the value of that same variable in the prior period, then the system assumes it cannot evaluate that formula for all periods simultaneously (which it normally tries to do). Instead, it groups these formulae together as much as it can, and places period-by-period loops around them. The second special case is more subtle. Suppose the user defines the following set of formulae:

- Interest expense is calculated from borrowings and interest rate.
- Net income is calculated from income before taxes, which is revenue less expenses (including interest expense).
- Retained earnings are calculated from net income (cumulated).
- Once the other balance sheet items have been calculated, borrowings (or cash on hand) fall out as the "plug" line that balances the assets and liabilities.

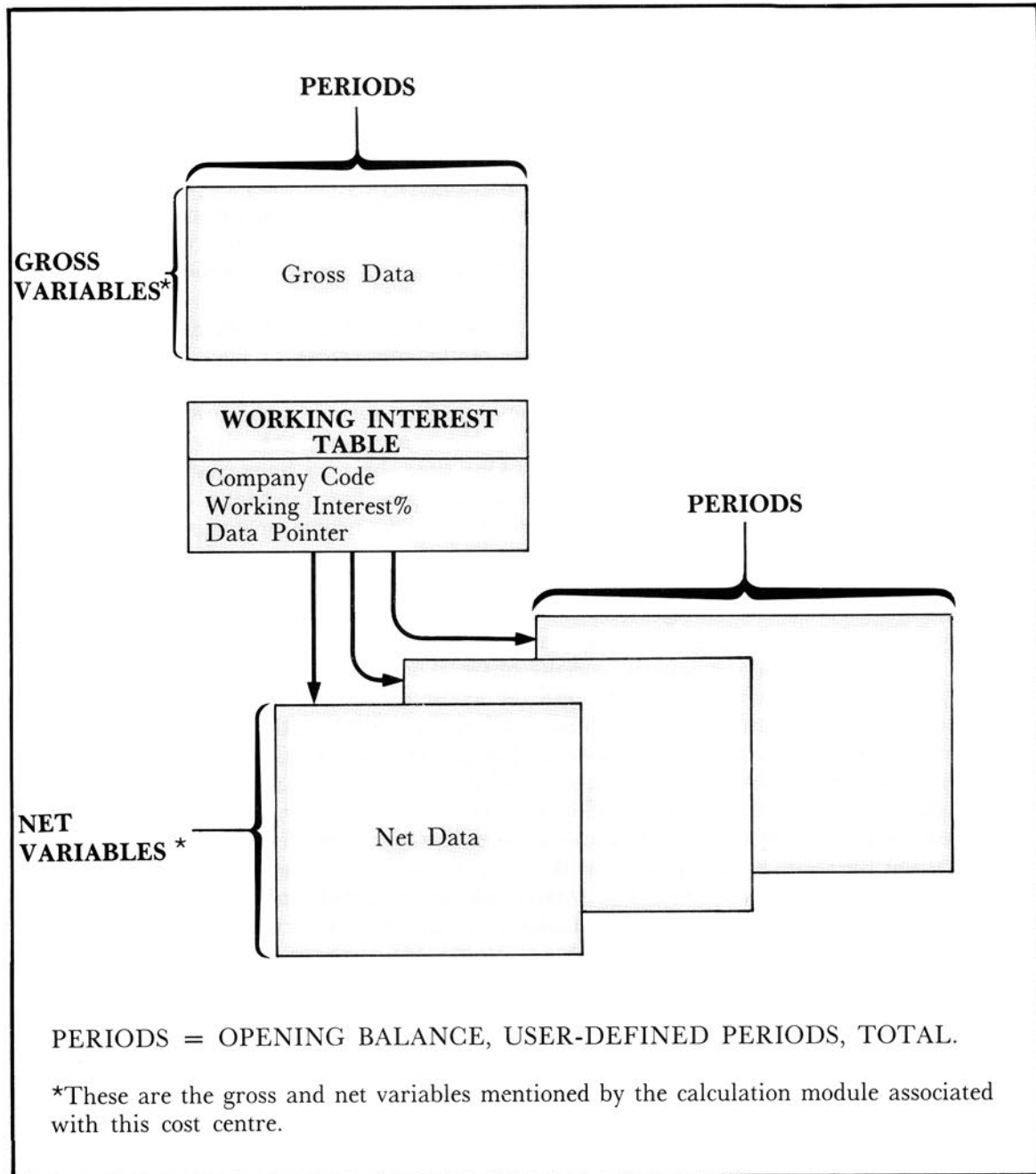
Here we have a chicken-and-egg situation in which interest expense depends (partially) on itself. The system recognizes such cases and places an iterative loop around the formulae involved. The loop repeats until convergence is achieved, unless the results are not converging after a finite number of iterations have taken place.

Cost centres are the hub of ABS. Each cost centre record holds the data for all variables associated with its calculation module, as well as a variety of other parameters concerned with the processing of the data. Cost centres are the focus of calculations and consolidations, and the source of data for reports.

A cost centre record is identified by a short code and contains a description of the cost centre. The most important parameter it contains is its calculation module code. This determines what data is held in the cost centre and how it is processed.

The data in a cost centre is held in matrices where the rows correspond to variables and the columns to periods. The system adds two columns to the data matrices, in addition to those required for the user-defined periods—a leading column for opening balances, and a trailing one for totals. Every cost centre has a data matrix for gross variables. It may also have matrices for net data (one for each company the user identifies as having a working interest in this cost centre). This organization is illustrated in Figure 3. The user implicitly adds or removes data matrices from a cost centre by adding or removing entries (columns) in its working interest table.

Figure 3
Organization of Data in a Cost Centre



Cost centres can be structured into a hierarchy, or a network, or a combination. A hierarchy of cost centres is defined by specifying the direct subordinates of each non-detail-level cost centre in the hierarchy. This can represent relationships such as oil

wells to fields to areas to regions to provinces, or departmental breakdowns within a company. The system accumulates data by variable as it consolidates upward through the cost centre hierarchy. A network of cost centres is defined by specifying inter-cost centre data transfers. An entry in the transfer table in a cost centre tells the system to take the data for a particular variable in this cost centre and add it into a (possibly different) variable in another cost centre. This can represent relationships such as those between liquids processing plants and storage caverns linked by pipelines. The user can define any mixture of cost centre hierarchies and networks as long as there are no circular data flow patterns.

The user may enter data into gross variables in any low-level cost centre (i.e., one that has no subordinates). If data is entered for a variable which is the result of a formula in the calculation module, the formula is not used. The user may enter data into several periods at once and has a variety of options when specifying the changes. He may enter new absolute values, add or subtract (in absolute or percentage terms) from the existing data, spread a total (evenly) across all periods, or enter starting values with inflation rates, deflation rates, or period-to-period absolute increments or decrements.

While the user can maintain his data in the cost centre phase, the process is cumbersome because it covers all other parameters of the cost centre as well as data. The system offers a data maintenance phase which gives the user more flexibility and less overhead for data entry. The user has all the capability offered by the cost centre phase at the point of data entry. But in addition he has control over the sequence in which the system prompts him for the key fields. One time, he may have changes to many variables in one cost centre and will ask to be prompted first (and least often) for the cost centre. Another time, he may have changes to one variable in many cost centres, so will instruct the system to prompt for variable first.

As the user sets up and changes calculation modules, cost centres and data, the system keeps track of where changes have occurred. It does not execute calculations until the user requests them by selecting the execute phase. The user may let the system execute every cost centre which has been changed (or whose calculation module has changed). Often, users will restrict the calculations to a few cost centres, either because these are the ones of current concern, or because he has just changed a calculation module and wants to verify the new formulae before recalculating all the cost centres that use the module. The user also has the option of doing the calculations immediately (as a T-task) or requesting a B-task, which—on weekdays—is scheduled to run after 8 p.m. When the calculations take place, the system automatically includes in the calculation list any cost centre that receives data from a cost centre already in the list. It also sorts the list so that each cost centre is calculated before any cost centre that depends on it. By this means, the effect of any change is propagated correctly through the database.

Many people judge a system by the quality of its output. The report generator in ABS offers the user extremely powerful capabilities for preparing a wide variety of reports. The basic concept behind the report generator is that a report can be thought of as an array with three dimensions (called zones)—its logical pages, lines, and columns. Data in the database is identified by three or four keys—cost centre, variable, period and (in the case of net data) subcompany. In theory, the report generator allows the user to match any zone with any key field (one zone may have two key fields to permit all four key fields to be used). In practice, however, a substantial amount of work is required to produce a set of *pro forma* or template functions for each combination. Only a handful of the most common combinations have so far been produced.

The user defines a report by entering lines of text containing keywords and arguments.

Most keywords and arguments are selected to be meaningful to the user. To the system, the most significant keyword is *SPECIFY*. A *SPECIFY* command marks the beginning of the definition of each zone of the report. Thus there must always be three of them in a report definition—one each for pages, lines and columns. For example, the command

```
SPECIFY PAGES = COSTCENTRES
```

marks the beginning of the definition of the logical pages, and states that each page will be derived from one cost centre or an arithmetic expression involving cost centres. Typically, this command would be followed by one or more *COSTCENTRE* commands, each one causing the production of one or more logical pages (which each may occupy one or more physical pages). In any zone, the user can use *EVALUATE* commands to perform arithmetic. Thus, a complete *PAGES* zone of a report might read as follows:

```
SPECIFY PAGES = COSTCENTRES  
COSTCENTRE DOME  
COSTCENTRE HBOG 'HUDSONS BAY'  
EVALUATE DOME + HBOG 'MERGED COMPANY'
```

(The third page could also be produced from a cost centre in the database whose subordinates are *DOME* and *HBOG*, if such a cost centre existed.) Of course, this example is only illustrative. It takes no account of the inter-company eliminations that occur during a merger, but merely adds together corresponding data from the two cost centres.

In the lines zone, the user specifies—in sequence—the lines that are to be printed on each logical page. If this zone starts with the command

```
SPECIFY LINES = VARIABLES
```

then each data line will contain values for one variable or the results of an arithmetic expression using the data for several variables. The user also indicates where titles, underlines, blank lines, etc. are to appear. All-zero data lines may be printed or suppressed; data lines for key combinations that are not present in the database are never printed. Output may be made dependent upon the appearance of the previous data line. For example, the sequence:

```
VARIABLE SUBTOT1  
DEPENDENT UCOLUMNS ; ;=
```

underlines all columns (except the first) on the line after the data for variable *SUBTOT1* is printed. If the data for *SUBTOT1* is all zero (or the variable is not present in the cost centre), neither it nor the underline is printed. Output can be made dependent upon the potential for print of data which is in fact never printed. For example, the sequence:

```
HIDE VARIABLE SPECPAY  
DEPENDENT LITTLE NB. SPECIAL PAYMENTS HAVE BEEN MADE
```

prints the note if the data for variable *SPECPAY* is non-zero, even though the data itself is not printed here.

Often the user wants to apply a single report definition to many similar, but not

identical, cost centres. The report generator simplifies this task in several ways. Dependent output is an example, as is the fact that the report generator ignores—rather than objecting to—requested variables that are not present in a cost centre. A third useful feature is the ability to hold output pending the appearance of later output. The system provides *hold buffers* where held output can be saved. Before printing any output the system first prints and clears the hold buffers. By setting the hold buffers at the proper time, the user can show titles above sections of a report which contain data, and suppress the titles when no data is printed.

The report function compiler has two passes. Pass 1 validates the user's specifications and builds internal tables. If the user is entering brand new specifications, pass 1 is done incrementally as each line is entered; thus the user is informed immediately if any error is found and can re-enter the line. When the user edits existing specifications the output from the text editor is processed by pass 1 to rebuild the internal tables. Pass 2 selects the appropriate set of pro forma functions and "compiles" them using the information in the tables. The output of this pass is a set of APL functions (usually about a dozen) which when executed produce the report specified by the user. These functions are saved with the report specifications for use when the report is run. Different sets of pro formas are needed for different report organizations because they need fundamentally different strategies to produce the reports with reasonable efficiency.

The implementation

Before discussing the implementation itself, I should describe the non-technical environment at the time I made my presentation. In 1979, the paperwork surrounding the creation of the 1980 budget had severely strained the capacity of the budget department. Since the company grew substantially between 1979 and 1980, the prospect of repeating the same process for the 1981 budget was regarded with apprehension. The Budget Manager quickly realized that what I was proposing would (if it materialized) alleviate many of his overload problems. He also planned to delegate to the Oil and Gas Division the handling of their own budget data using the system. Thus he became a strong and knowledgeable ally on the few occasions when a conflict had to be resolved. I was also fortunate to receive the full support of management in my own department, even though this was the first major APL system they had been involved in and the time scales I was proposing were ridiculously short by their standards (using "traditional" systems development techniques).

Because of his deadline to present a finished budget, the Budget Manager insisted I must have something to show by the beginning of July. Obviously, I could not produce the entire system by then, so we planned a staged implementation.

The first stage provided the capability to maintain variable definitions, calculation modules, cost centres and data and to perform the calculations. This was released at the beginning of July and the users immediately started entering the specifications which they had been preparing while I was writing the programs. The process of defining and refining their specifications kept the users productively busy through July while I proceeded with stage 2 (in between fixing the bugs that emerged from stage 1).

The users had the ability to do (almost) everything they needed with the system, except produce tidy reports. (They could produce simple listings of all their specifications and data for verification purposes.)

For stage 2 (the report generator) I became almost a hermit for several weeks while I wrestled with numerous designs and redesigns. The major problem was to structure the internal tables so that (a) they could be built from the user specifications, and (b) they would have enough flexibility to accommodate report formats (pro formas) which we would write in the future. Of critical importance were the many decisions on when particular actions could take place during the report generation process and on which combinations of parameters they had to depend. Many alternative designs were examined and advanced to various stages of completion, only to be scrapped or reworked when they failed to handle all combinations of specifications. Eventually a design emerged which—so far—has provided all the necessary flexibility. The first pro forma (*PAGES = COSTCENTRES, LINES = VARIABLES, COLUMNS = PERIODS*) was released in August, just in time to produce the budget reports. The second pro forma (with *LINES = SUBCOMPANIES, VARIABLES* to permit the display of net data) followed quickly. This completed enough of the system to handle all the requirements of the 1981 budget.

The last stage of the initial implementation was to transfer the budget data out of the APL database into the main accounting system. Having anticipated this need, I had built into the cost centres tables which identified where in the accounting system each piece of data had to go. After the report generator, this part of the development was easy. All I had to do was extract the appropriate data and keys and format them into transactions which would be acceptable to the accounting system edit programs.

Exploiting the system

One Friday morning in December 1980, I received a phone call from the Controller. He was trying to do some financial analyses, but having difficulty coping with the plethora of alternatives that were being considered and the volume of calculations, which his analyst was currently doing by hand. Could I help? Thus began the exploitation of the system for analyses outside the budgeting area.

The problem at hand was to examine various ways of structuring the financial setup of Dome Canada. I was confident that ABS could eliminate the calculation problems. By Friday night, the analyst and I had built a crude model of the situation. On Saturday we added a number of refinements and I began to show the analyst how to use the system. By Tuesday we had a good working model and the analyst was using the system confidently. The crowning point came when we were able to take into a meeting reports from the system incorporating major changes to the data which we had received just thirty minutes previously.

Since then, there has been no looking back. There are now over a hundred user-defined ABS databases in existence. Almost all of them are set up and run without any involvement of systems staff. A number of departments have chosen to process their own budgets in separate databases in great detail, thereby eliminating almost all the manual calculations associated with budget preparation.

The financial planning and forecasting departments have set up numerous models to analyse various situations, either as one-shot exercises or on an ongoing basis. Usage of ABS forms a significant percentage of the total APL workload. We see distinct peaks in the load, which coincide with important events, such as the establishment of Dome Canada, acquisition of HBOG, Federal budgets, etc.

The system has indeed succeeded in its primary aim—to make the power of the

computer available to our business analysts without requiring of them any programming knowledge. Many people have indicated they now feel in command of the numbers they process, rather than slaves to their calculators. Users themselves define the rules of the game, and re-define them as the need arises. When the National Energy Policy was announced in October 1980, users in the Oil and Gas Division told the system how to calculate and account for the new taxes. They were producing a revised set of budget reports the following evening (and to this day I've not seen what they did!).

Extensions

After a system is implemented, one inevitably realizes that it could be improved. Fortunately, before I joined Dome I had written similarly styled (though much simpler) systems. Thus, the initial implementation of ABS was fairly robust, though there were many constructive criticisms from the users, most of which we responded to. For example, if a user enters a formula in a calculation module that mentions an undefined variable, the system now offers the ability to define the variable at that point, instead of forcing the user to go to the variable definition phase to add it before he can enter the formula.

While considering suggestions for enhancements, we are always very conscious of the trade-offs among usefulness, generality and implementation feasibility. Some changes were useful, general and easy to install. These we often did very quickly. At the other end of the scale, we have had requests that were very application-specific or simply impossible to provide with reasonable effort. Many of these have not yet been installed; some never will. But even here, we have managed to satisfy some requests, or parts of them, in more general features.

The major enhancement to ABS has been the multiple-case ability. In the original version of the system the user was allowed to have only one set of data and parameters for each cost centre in a database. As ABS was used more for planning, it was obvious we had to allow multiple copies (which we call cases) to handle what-if questions and alternative scenarios. Within a cost centre, each case has its own calculation module, subordinate cost centre list, partner working interests, and of course data. Each case (except BASE) has a parent and the system automatically uses the closest ancestor of a case if the case it is looking for is not present in a particular cost centre. Thus, the user need only specify new data in cost centres where it is different when defining a new case. Parameters and data which the user doesn't change are inherited from the parent case. This feature has been extremely popular with users in the forecasting area. Some of their databases have more than 50 cases (or scenarios) defined. As a matter of interest, this feature uses—indeed, is made practicable by—general arrays. Each cost centre used to be a package containing the parameters. Now it contains a two-dimensional general array, each row holding the parameters for one case. Thanks to coadunate representation, we can hold many cases in one cost centre record without incurring *WS FULLs*, because many of the parameters (the larger ones especially) are generally the same from case to case.

One impact of the introduction of multiple cases was an adverse one on the ability of users to understand all the whys and wherefores of the system. The case ability brought with it a host of new defaults and rules which the system had to follow. Thus we saw a dramatic increase in the number of occasions when users became confused about how a particular value arose. We responded in two ways. First, we gave the user greater ability to take quick snapshots of sections of the database. This did nothing

to help the user discover where a particular value came from—it just showed him what the value was. Our second move in this area provided a diagnostic facility which attempts to explain to the user precisely how a particular value was derived. For example, it distinguishes between values that are constants, input data, results of user-defined formulae, consolidations from lower-level cost centres and data transferred in from other cost centres. If the diagnostic facility discovers a discrepancy which it cannot explain, it admits the fact and suggests to the user what he might do to correct the problem. Needless to say, this facility has very quickly become popular with ABS users who have complex databases.

Two other enhancements which quickly became popular with our users were the *NEWS* and *GRIPES* features. *NEWS* allows us to keep users informed as we make changes to the system, or to disseminate information to help the users. *GRIPES* allows users to comment (favourably or unfavourably) on the system and to request changes. Users other than the originator can endorse a gripe and thus lend their weight to the originator's viewpoint. These two facilities allow users to feel more actively involved in the development of the system.

Another feature which we have found useful is a global trap in the ABS workspace which saves any otherwise unhandled event in a file. This feature is invisible to the user—he still sees the normal event report. But because we examine the event file regularly we can discover and fix problems more reliably than if we depended upon users to report them.

There have, of course, been many other minor enhancements and improvements to the system. Some were implemented to fill gaps in the system's capabilities. Others increased the number of requirements the system was capable of meeting. In some cases, an enhancement requires a change in format or content of user databases. Since our development account does not have access to most of them, the system has an automatic upgrade capability. When a database upgrade is required, we write an upgrade function and tell the system to use it to upgrade all databases. The next time a particular database is accessed, the system will execute the upgrade function to make the required changes. This has allowed us to install many features, yet have the system handle the chore of keeping user databases up-to-date.

Retrospection

In many senses, the development of ABS began long before I joined Dome. For many years, I have felt that the best way to help planners work more effectively is to put into their hands the power of the computer to organize and process data and prepare reports. This leaves the planner free to do what only people can do (so far)—to think of the right questions to ask and work out the implications of the quantitative outputs from the analyses.

When I started using APL (after ten years' experience with many other languages) I quickly realized it was a tool with enough power to support the development of very flexible systems. Gradually over the next years, I built up an extensive library of utility functions. With each new project, I took advantage of this library, and the ease with which one can use plug-in modules in APL, to reduce the amount of new code I had to write. An invariable spin-off from each new system was a few more utilities which were documented and saved in my library for future use.

Figure 4
ABS Calendar of Events

DATE	EVENT	DEPARTMENTS USING ABS +	USER CONNECT HRS *
1980			
May	ABS development started		0
July	System released to users (without report generator)	Budget, Oil & Gas	490
August	Report generator released to users		250
September	1981 budget preparation complete		370
October	NEP announced; users revise 1981 budget		220
December	First non-budget use (Dome Canada analyses)	Controller	510
1981			
February	1981 budget transferred to accounting system		490
Spring	Models set up to analyse long-term debt and HBOG acquisition	Financial Planning, Finance	550
June - September	1982 budget preparation	Data Systems	790
November	Budget revised in response to federal budget		720
December	Development of multiple case facility started		1,000
1982			
January	1982 budget transferred to accounting system		1,190
	Five-year forecast developed using ABS	Forecasting, Economics	
March	Multiple case facility released to users. Forecasting department begin to analyse numerous (> 50) scenarios using ABS		1,590
May	Users given ability to combine multiple calculation modules into one module		1,710
June	PEEK, NEWS, GRIPES, DIAGNOSE and automatic event recording implemented.		
+ cumulative list			
*approx. monthly connect hours by non-systems staff using ABS			

By the time I joined Dome, I had already developed or been involved in several systems which exhibited (to a lesser degree) some of the flexibilities of ABS. At Dome, I realized the users were desperately in need of a system such as this; moreover, they were typically the kind of user who could appreciate and use it effectively. At the same time, I felt I was ready to pull together my past experience and build such a system.

In the past two years, the system has indeed flourished as shown in Figure 4. In large part this is due to the fact that the environment was ripe. More than a third of all users on our in-house system now have access to ABS. Many of them use ABS regularly, and many use only ABS. I have heard users claim that they feel in control of the computer, rather than at its mercy. On the other hand, I have seen a few users who never really grasp what the system is all about. Ability with the system seems to be more a question of attitude than anything else. A secretary may find she can make the system jump through hoops, while her boss flounders.

All in all, the past two years have been a tremendous challenge. First there was the problem of putting the whole system together and getting it to work (especially the report generator). Then there has been the task of keeping the system running and competitive in an environment where users are aggressively stretching and twisting it and are never afraid to voice their opinions when it does not meet their demands, or breaks at the wrong time (i.e., any time).

The success of ABS is due in no small part to the users. They have not shirked their responsibilities (a) to ensure their needs are reflected in design proposals, and (b) to accept the role of guinea pig when new features are first released. One lesson I have learned from ABS is that such a system can succeed only in the right user environment. At Dome, the need was urgent, the users were willing and I was free to develop the system without the nit-picking and political manoeuvring that often accompany such an exercise.

Acknowledgements

I owe a debt of gratitude to John Macklem, Manager Budgets. From the beginning, his influence was instrumental in helping me steer the straight and narrow between the rocks of inadequate or inappropriate designs and the stormy waters of over-ambitious ones. He was quick to grasp the significance of what we were trying to do and became a source of inspiration. He also acted as mediator between the demands of the users and what is technically feasible.

I also must thank the members of my own department who have provided an environment (technical and managerial) conducive to this development. Edward Keyser, who joined Dome early this year, has picked up the internals of ABS with great enthusiasm. His effort has allowed us to speed up the enhancements and serve our users more promptly.

Reference

Martin, James. *Application Development without Programmers*. New York: Prentice-Hall Incorporated, 1982.

TOP DOWN BUDGETING: A FAIR-SHARE APPROACH

Vincent J. Palese, Jr.
Manager, APL Product Support
General Services Division
Xerox Corporation
Rochester, New York

Introduction

The Xerox Corporation is a major user of SHARP APL. The fast program development time, existing telecommunications networks, and easy availability of computer power are the basic qualities that make APL well suited to the business requirements of the end user community.

This paper will describe an APL Planning Model used by Xerox's Field Financial Operations Group. This Group comprises individuals with varying degrees of APL competence. Given the different skill levels, the Planning Model must be easy to use. In a short planning cycle, analysts need to spend time on business logic and equitability analysis, rather than the technical aspects of the computer environment. Also, the model is designed to be flexible and responsive to the dynamic nature of market-place planning.

Background

In the early to mid-1970's the process for developing Sales Branch plans consisted of a "top down" allocation methodology. Then, as now, Headquarters was responsible for creating individualized Sales Region plans. Tape copies of specific Region plans were sent via air-freight to each Region headquarters. The data was then loaded to disk on a mainframe located at the Region site. Region staffs were responsible for developing their own allocation methodologies and translating these methodologies into programmable code. The entire process extended over a three month period, including agreement on methodologies to be used locally, the implementation and testing of the computer programs and the actual allocation process itself. The methodologies were based on historical data and market-place potentials. Finalized Branch plans were sent to Headquarters via tape.

In 1977 a more sophisticated Planning Model based in SHARP APL was designed and implemented. Since that time, a number of enhancements have been added, particularly in the areas of methodology refinement, analytical reporting, cost-savings techniques and general system maintenance. The model provides a consistent framework

of methodology, file structure and analytical reports which increases the year-over-year productivity of the Planning Group.

Business requirements

- 1) Allocate a Division level marketing and revenue plan to all Sales Region and Branch locations.
- 2) Allocate full year plans by product line and activity factor. Activity factors include both "Primary Factors" (sales transaction types) and "Secondary Factors". Secondary Factors are not allocated but derived from the primary factors. Secondary Factors usually maintain the same share of the total plan as the Primary Factor used for the derived factor calculation. An example of a Secondary Factor is Sales Revenue, which is the Primary Factor "Sales Installs" multiplied by a price list.
- 3) Develop individualized methodologies for each unique activity factor. The methodologies used will be consistent for all Regions. Different members of the Planning Group are responsible for the different factors. This responsibility includes developing the allocation methodologies, writing and/or maintaining the allocation programs, and analyzing the resulting plans.
- 4) Use historical performance and expected market-place potential for the plan allocation. Different history/potential weightings will be used to test the sensitivity of the Region shares. The final methodology used for each factor is expected to ensure an equal opportunity for each location to achieve its full year plans.
- 5) Incorporate into the "business-as-usual" base case the incrementality and product diversions expected to result from the introduction of new products and/or strategies planned for the coming year.
- 6) Have the ability to finalize resulting full year plans at the Region level by the Headquarters Planning Group. Straw man Branch plans are released to the Regions for "fine-tuning". By doing this, Region management can exercise their knowledge of local market-place conditions and ensure plan equitability among their Branches.
- 7) Calendarize each activity factor (both Primary and Secondary) to reflect the same calendarization as the initial Division Plan. Calendarization is completed after senior management has finalized the full year plan shares for all locations.
- 8) Have the ability for Branch and Region Plans to be used by other planning and operations groups: Territory Sales Rep Budgeting; Supply/Demand Planning; Expense and Resource Planning; and Equipment Planning and Control. Therefore, the planning system must be flexible and easy to understand by many downstream users.
- 9) Have the ability to easily modify the Model. Base cases, new product cases, organization structures, and market-place strategies are often modified during the planning cycle. The allocation model must be able to respond quickly to these basic input and methodological changes.

System design

As previously stated in the business requirements section, the planning process for

Xerox marketing Branches is a "top down" process. To put it simply, the Forecasting Group gives the Planning Group a "pie" which must be carved into equitable slices. The planning process involves two different types of allocations: Primary and Secondary (Derived) Factors. Considering all months and products, the initial allocation of the Primary factors results in 1.2 Million slices of the "pie". These primary allocations are then used to derive the Secondary Factors which account for an additional 3.5 Million "slices".

The purpose of noting the mass of detail required in this planning process is to emphasize the data manipulative power of APL. The data structure of the initial allocation is a matrix of products as rows and cases and columns. Cases include the "business-as-usual" base case for existing products and additional cases for new products and marketing strategies to be implemented in the coming operating year. At this point, cases represent full year plans. Basically, any case which requires a unique set of conditions is treated as a column in the initial full-year allocation matrix.

The allocation stage for each Primary Factor begins with a vector of numbers representing product line detail for an individual case. The allocation methodology for each factor explodes the vector into a matrix of product lines by locations (Regions/Branches). The sum of the locations for each product equals the original vector set. By having each factor in its own matrix it becomes easy to modify each individual activity factor without impacting other activity factors or other cases.

This is accomplished by a basic utility function, known as *FACTOR*. This user-defined function is the key driver for each of the technical implementations of the allocation methodologies. *FACTOR* is a dyadic function which returns an explicit result. The left-hand argument of *FACTOR* is a scalar integer or vector of plan numbers (integers) to be allocated. The right-hand argument is a numeric array, whose rank is one larger than the rank of the left-hand argument. This array represents a weighting scheme to be used in the allocation of plan numbers. Figure 1 illustrates an allocation of 202 units. It makes no difference to the *FACTOR* function what the order of magnitude of the weights are. For example, weights of 20 22 25 26 24 30 (see column A) have the same meaning as weights of 0.2 0.22 0.25 0.26 0.24 0.3. *FACTOR* determines the share of each individual weight as a function of the sum of the weights (see column B). The plan numbers are initially allocated based on the individual weight share. Rounding does present a problem (see column C), since the plan numbers can only be allocated as integers. To provide consistency in results from allocation to allocation the rounding technique used is to distribute the residuals to the intermediate (rounded) allocations, where the addition of the residual would have the least impact (see column D). That is, residuals are assigned to the intermediate allocations based on the descending order of the intermediate allocations. The final plan allocation, with location 6 receiving the residual unit, is shown in Column E.

Figure 1

	202	FACTOR	20	22	25	26	24	30
	(A)		(B)		(C)		(D)	(E)
	WEIGHTS		WEIGHT SHARE		UNROUNDED ALLOCATION		INTERMEDIATE ROUNDED ALLOCATION	FINAL ALLOCATION
	-----		-----		-----		-----	-----
LOCATION 1	20		.13005		27.482		27	27
LOCATION 2	22		.14966		30.231		30	30
LOCATION 3	25		.17007		34.354		34	34
LOCATION 4	26		.17687		35.728		36	36
LOCATION 5	24		.16327		32.981		33	33
LOCATION 6	30		.20408		41.224		41	42
	---		-----		-----		---	---
	147		1.00000		202.000		201	202

As defined previously, the allocation process is working on the "business-as-usual" base case and the new product/strategy cases. The base case allocation uses both historic and market-place potential inputs. The type of potential data and the relative weightings of the history and potential will vary by activity factor. Historical data used in the allocation is taken from Division data bases which contain monthly history by product line and various activity factors. During the processing of an allocation workspace, a program, *READ*, is used to retrieve historical data from the Division data bases. Historical information is not stored in the Planning Model, thereby eliminating the cost of storing redundant data. Since the planning cycle extends over a four month period, the Planning Model prompts a user for the latest month of year-to-date actuals to be retrieved. This provides for an allocation which includes the most recent available history (N months). The number of months of historical trend used depends on the methodology recommended by the Planning Group. The *READ* function uses two tables which map the historical product line structure and the activity factor definition of the historical data base to the Planning Model. This is a necessary process, since new products may be added to the line, old products may be retired and new sales strategies may be developed which create new activity factor definitions.

In contrast, new product/marketing strategy cases do not have their own historical base to use for the allocation. Therefore, market-place potential usually becomes the sole determinant. New products and new marketing strategies each have an overlay case. The product line detail for these cases includes both the plan for the new product and diversions for the products affected by the introduction of each new product. The net result for each case is the incrementality to be gained. After the overlay cases are allocated, the base case is adjusted by adding the new products and subtracting the diversions for the old products for each overlay case. The net result is the final full year product line plan (see Figure 2) of 500 units.

Figure 2

	<i>BASE CASE</i>	<i>PRODUCT 2 OVERLAY</i>	<i>PRODUCT 5 OVERLAY</i>	<i>TOTAL PRODUCT PLAN</i>
	----	-----	-----	-----
<i>PRODUCT 1</i>	100	(25)	0	75
<i>PRODUCT 2</i>	0	100	0	100
<i>PRODUCT 3</i>	100	(25)	0	75
<i>PRODUCT 4</i>	100	0	(25)	75
<i>PRODUCT 5</i>	0	0	100	100
<i>PRODUCT 6</i>	100	0	(25)	75
	---	---	---	---
	400	50*	50*	500

* Net Incrementality

Although there are different ways for handling new product/strategy plans, we have found that the consistency of this procedure simplifies the downstream planning and analysis process. In other words, the new products become part of the total planning process and are not treated separate and apart from the system. Since new products/strategies may be designed or targeted for specific geographical areas, it becomes easier to analyze their bottom-line impact on the total plan. This is especially critical as assumptions for new products/strategies are evolving and do change during the planning cycle. The system has the capability to incorporate these changes on both the base and overlay cases. This capability involves the creation of new plan "versions" which can include different combinations of products, new product launch schedules and strategy implementation.

FACTOR is one of many tools (approximately eighty user-defined functions and variables) available to users of the Planning Model. These tools are stored as "Packages" in an APL file rather than in the allocation workspaces. Use of "Packages" has simplified the maintenance of the Model for at least two important reasons. First, access of the tools from a single source ensures that the workspaces use only the most recent version of a tool. This avoids the problem of having to load each workspace, copying the most recent version of a tool into the workspace, and then saving it. Second, this provides an "uncluttered" workspace for an end-user to maintain. User-defined functions in a workspace are limited to those functions directly related to the allocation methodology. System maintenance is simplified, as the function listing for the workspace contains only the functions used for an allocation. This can, for example, reduce the function list from ninety user-defined functions to ten. This is advantageous to a user, since debugging any problems is easier with ten functions to review than it is with ninety.

System requirements

The Planning Model

- 1) The model uses three basic data inputs: the base/overlay cases to be allocated, historical data, and market-place potentials which are used in the allocation process (see Figure 3). These inputs are loaded into the Planning Model via feeds

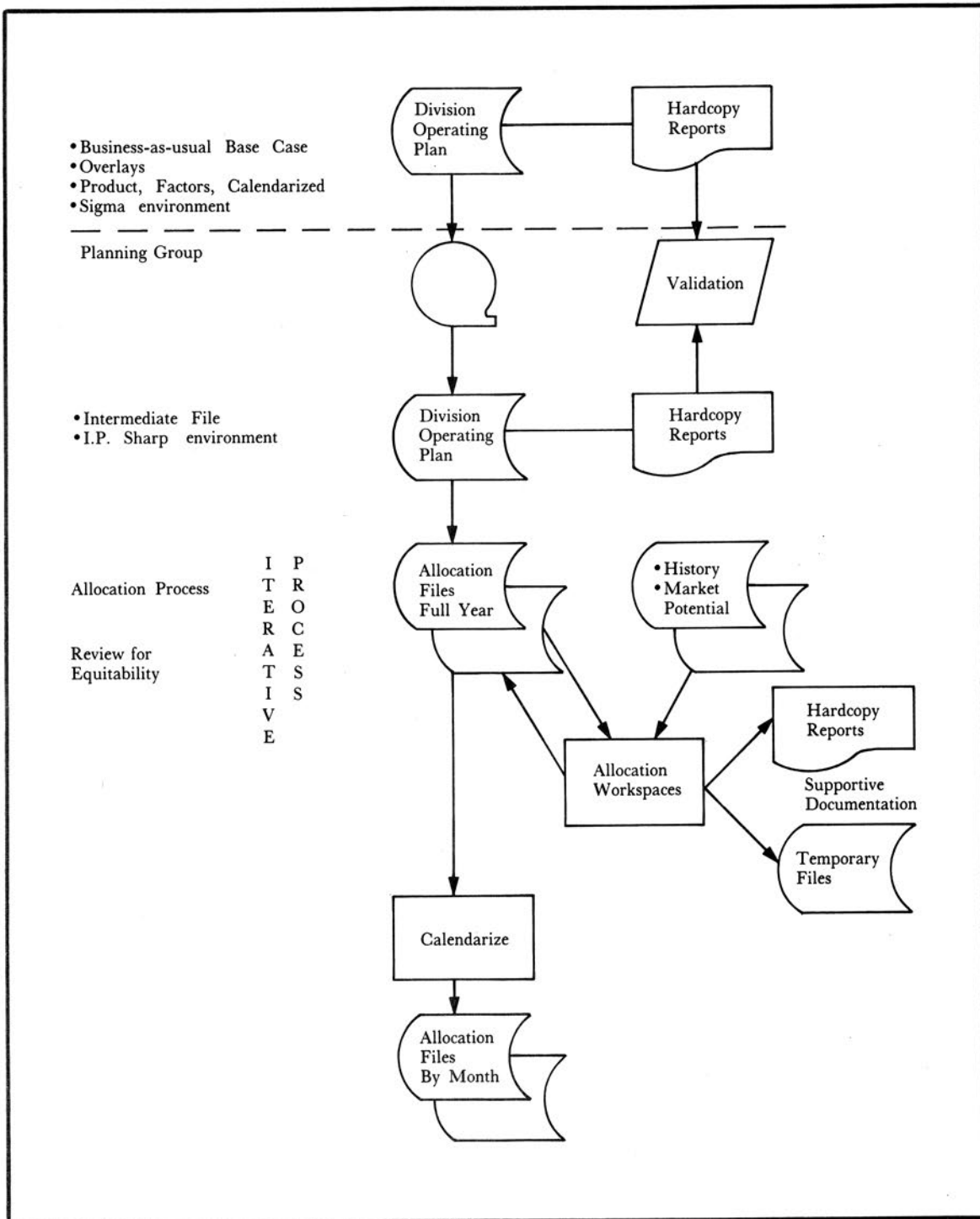
from other data bases or manual entry. The Planning Model resides on the same mainframe and in the same environment as the basic data inputs. The validation process involves both the Planning Model administrator and individual members of the Planning Group. A table is used to map the historical data bases product line structure to the product line structure of the Planning Model.

- 2) The model comprises a group of workspaces. Each set of related factors has its own workspace. Different members of the Planning Group are responsible for developing the allocation methodologies for the different factors. This responsibility also includes the programming and maintenance of the workspace(s) which perform the allocation, and the development of files or hardcopy formats for analyzing and reporting the results.
- 3) Members of the Planning Group develop the allocation methodologies which are reviewed and approved by both Headquarters and Region management. Although the methodologies differ for the allocation of each Primary Factor, they all require the basic data inputs of history and/or market-place potentials. Selection of potentials is based on the predictive reliability and the availability of the data. The Planning Model allows for user input of different weightings for the history/potential relationship. Resulting allocations, based on different weights, are reviewed by management. Although the Planning Model may provide the best mathematical answers, there may be unique market-place events which are not easily captured through the data supplied as an input to an allocation. To put it simply, technical tools are not substitutes for business judgement. For this reason ease of changing data in response to knowledge of the market-place by Headquarters and Region management becomes a prerequisite of a meaningful planning process.
- 4) Each workspace is a "stand-alone" segment of the total Planning Model. When a change is needed to either the basic data inputs or the allocation methodology, it is necessary only to modify a specific workspace. This simplifies the testing and review process. For instance, if problems arise with one particular factor, the efforts of other members of the Planning Group working on the allocation or analysis of other factors need not be affected. This eliminates any "bottlenecks" which could result from the use of multifactor allocation workspaces. APL is suited for the "stand-alone" processing requirement, as it allows for multi-user access of the same basic input and output data files.
- 5) Individual workspaces can be "chained-together" to allow all allocation workspaces to be executed as a single job stream. This includes both the allocation of Primary Factors and the calculations of Secondary (Derived) Factors. This job stream can be run in either a T-task or N-task mode. Because of the volume of analytical reports that support each workspace, the N-task option provides both a cost-savings due to the CPU-discount given to N-tasks, and a faster turn around of the results. The benefits of this single job stream are the preservation of data integrity, through the proper sequencing of all Primary to Secondary calculations, and the time-savings that result from a system-controlled rather than a user-controlled operation. In other words, analysts don't have to "terminal sit." The job stream has a built-in monitoring tool which informs a user of the progress of the task. Once the allocation of a factor in the workspace chain has been processed, a user can begin analyzing the results, while the remainder of the job stream is still in the process of being executed. This increases productivity of the

Planning Group, since time is not wasted waiting for the entire job stream to complete before the results can be analyzed.

- 6) The Planning Model uses a limited number of programming conventions which have to be adhered to in the technical implementation. This is a human-design consideration. Planning Group members need to devote their time to analyzing the equitability of the results, not convention-following. The only conventions which must be followed are: (a) use of three "reserved" variable names which identify the files to be updated. These names indicate the Region, Historical month, and the Case Version number; (b) use of three user-defined functions which provide the capability for chaining workspaces together and accessing the proper input/output data files; and (c) the reservation of a specific file tie number for an analytical report output file. This number serves as a signal to the system that workspaces are chained. It is based on the fact that, in APL, files remain tied to a specific tie number during a task.
- 7) The entire planning cycle extends over a period of three to four months. This includes preliminary analysis of alternative methodologies, preparation of the Planning Model, review of potential new product/marketing strategy cases, and the final allocation process itself. The allocation process is compressed into six weeks. Given the short timeframe required, approval of the final Region allocations by Headquarters management requires the immediate release of the plan files to the Region staffs for their review. To facilitate this process test cases are released to the Regions during the planning cycle. The test cases serve to familiarize the Region staffs with the allocation methodologies, the supportive analytical reports and also allow for a preliminary review of the expected results. Since Sales Regions and Branches are geographically distributed throughout the United States, a communications network allows immediate access to the plan files. File access for each Region is limited to only their Sales Branches. The SHARP APL file subsystem is ideally suited for the timely release by Headquarters and the immediate access to the plan files by the regions.
- 8) In "top down" allocation, the sum of the finalized Branch plans must be equal to the initial Division plan for each product, month and factor. Tools to validate this requirement are provided. These tools facilitate immediate validations which must be completed before any analysis of the results is undertaken.
- 9) Canned and variable format reporting is built into the Planning Model to aid in the analysis of the results. Plan allocation shares by location and factor are compared to history. Several different weights can be run on the same "version." This simplifies the selection of the final methodology.
- 10) As stated previously, all allocations are performed on the full year division plan. A separate workspace is used to calendarize each location's full year plan for each product and factor. The calendarization is done first on a quarterly basis, and then by months within each quarter. The final calendarization plan for a location reflects the same month-to-month trend as the initial Division plan. Validations are provided to ensure that no negative numbers occur where not appropriate and that the full year totals in the calendarized and non-calendarized allocations do not differ.
- 11) The resulting calendarized plans are provided to the Territory Budgeting Planning Group. In turn, a similar "top down" allocation process is undertaken by this Group.

Figure 3
Planning Model Overview



THE DESIGN AND IMPLEMENTATION OF AN INTEGRATED COMPUTER SYSTEM IN A VETERINARY COLLEGE

**B.W. Stahlbaum
and
A.H. Meek
Ontario Veterinary College
Guelph, Ontario**

Background

As veterinary services become more concerned with ecologically complex problems, the approach to health care delivery must be based on more comprehensive information, and on continuously upgraded decision-making skills. The primary objective of an animal health information system is to provide data which will assist decision-making both with regard to existing programs and practices, and for the planning of new ones. Better information does not necessarily make management easier; rather, the manager who is better informed needs to be more skilled at analyzing data, and more innovative in interpreting it.

An information system must provide for: (1) the collection, recording, and storage of data, (2) the retrieval, analysis, and interpretation of these data, and (3) the rapid and regular communication of the information to those who have the responsibility and authority to act.

Data put into such a system may arise from a variety of sources; for example, livestock owners, private and public practitioners, diagnostic laboratories, and research and teaching institutions. It is to the needs of the latter source, and more particularly the Ontario Veterinary College (OVC), that the project described herein is addressed.

The Ontario Veterinary College of the University of Guelph is one of three Canadian schools that offer degree programs in veterinary medicine. In February of 1980, OVC management decided that an integrated, computer-based system should be developed to aid its departments in developing an overall system to function in teaching, research, hospital, and extension activities.

Within the OVC, the information system is intended to have application both in the medical area *per se*, and in the administrative/business area. In the medical area, the information management system should be a prime factor in motivating and guiding:

- (1) Researchers—in establishing research priorities, in developing (and extending) new knowledge, and in planning, monitoring, and evaluating health management services.
- (2) Teachers—regarding the development and execution of teaching programs.
- (3) Hospital staff—in the provision of patient care and in the delivery of support services, e.g., diagnostic laboratories.

In the administrative/business area, data processing and information needs can be separated into those concerned primarily with the teaching hospital, and those of the College and Departmental administration. In general, the needs of the teaching hospital are the same as those of any business, and include fiscal management among other things. In this regard, charge capturing has been viewed as a logical by-product of the system, not as an end in itself.

In order for the system to be effective in the OVC environment, a number of other objectives are involved:

- (i) To develop a responsive system which will increase the efficiency and speed of information flow, and hence allow faculty and staff to increase productivity.
- (ii) To extend the present capabilities of faculty by allowing them to take advantage of the wealth of veterinary medical information and experience that is currently not readily available, and to reduce the unrealistic dependence which is placed on human recall of medical knowledge.
- (iii) To minimize the time-consuming, routine clerical duties associated with patient care and ancillary diagnostic services.

Management

The essence of successful project management, in an undertaking of this magnitude, is the effective interweaving of the responsibilities performed by everyone involved with the project. The three groups (within the infra-structure of the OVC) involved with the computerization endeavour are the Computer Group, the Computer Policy Committee, and the Area Managers (one from each of the almost two dozen functional units within the College).

The Computer Group (composed of a Coordinator, a Staff Analyst, several Analyst/Programmers, and a User Services Assistant) is the project's management team and driving force. Through their involvement and direction, all aspects of the project's justification, design, development, implementation, and support are undertaken. They initiate and coordinate the activities of all areas contributing to the project. They provide the project's control and thus work closely with the College's Computer Policy Committee and Area Managers.

The OVC Computer Policy Committee (composed of the Associate Dean, four department chairmen, four OVC area representatives, and two other University resource persons) provide the College's overall perspective to the project with regard to policies, priorities, and project review.

The Area Managers represent user management and work through their respective department chairmen to provide the necessary details for the Computer Group.

Since the system being developed is in response to a demonstrated need, and since the implementation may result in changes to an existing mode of operation, user participation in the project is mandatory. Users contribute to the setting of project objectives, and ensure that they are met. They evaluate, actively participate in, and reach agreement on various aspects of design, planning, implementation, and training which affect their own area. Extensive user involvement is necessary if optimal use of the system is to be made once it is in place.

Phases

The following phases of development have been completed:

I Project initiation

- commenced August 1980; completed October 1980
- defined the problem (as discussed above), its causes and consequences (in general terms), and presented a strategy for implementing the system

II Preliminary external design

- commenced November 1980; completed January 1981
- developed the system requirements to a level of external design sufficient for understanding "from the outside"
- evaluated its acceptability for use
- developed general plans for the balance of the project

III Feasibility

- commenced February 1981; completed June 1981
- evaluated the operational, economic, and technical feasibility of the system
- provided time, cost, and personnel estimates for various alternative solutions
- recommended proceeding to the next phase, hiring staff, using SHARP APL, acquiring various equipment, and conducting regular project reviews

IV Final external design

- commenced August 1981; completed March 1982
- completed the external design for functions identified in the preliminary external design
- confirmed the project development plans

The remaining four phases are repeated for each functional area (e.g., radiology, bacteriology, etc.) of the College.

V Internal design

- produces detailed system specifications
- extends the project development plans by identifying, describing, prioritizing, and scheduling the next stages

VI Development

- analysis and design of the next stage of development according to the detailed internal design document

VII Implementation

- programming and testing through active participation by managers, users, and developers
- installation and acceptance of the portion completed

VIII Post implementation review

- measures the degree of achievement of the objectives of the system
- evaluates the performance of the system with regard to estimated costs, performance, and user acceptance
- plans for future monitoring and evaluation

Internal design began in the Admitting and Medical Records area in May 1982, with implementation (i.e., "production" or full-time use) scheduled for September 1982. Other functional areas will be phased in as is feasible over a three-to-five year period. In this regard, and in order to help ensure end-user participation and acceptance of the system, it will be introduced and developed in an evolutionary modular fashion. That is, even within functional areas, a conscious decision has been made to start with small sections and then, as experience is gained (by both users and developers), to add additional sections and/or enhancements. These phases exist:

- (1) to provide a means of evaluating the project's progress within the College
- (2) to ensure a timely flow of information concerning the project to the areas affected by it
- (3) to ensure that all areas have an opportunity to review and to approve aspects which affect them
- (4) to ensure that OVC administration is kept aware of the status of the project

Overview

The primary goals, from a functional point of view, of the OVC system are:

- to collect and maintain accurate, comprehensive records for all patients
- to produce management and administrative reports for the College, the Departments, each of the Functional Areas, and the Hospital
- to communicate information for the purpose of assisting with patient care, teaching, research, extension, and College activities

The amount of information to be collected is determined by each functional area within the guidelines of the Hospital administration. This will obviously depend on case loadings and the types of problems that occur; however, the following figures can be used to approximate the volume of information that may potentially be gathered by the College:

- the Teaching Hospital has 500 “beds”, i.e., stalls/cages
- there are approximately 30 admissions and discharges per weekday
- each case (whether a new admission or a re-admission) is expected to generate (on average) 15×“words”, 5×“phrases”, and 2×“sentences” of required “electronic” information during its visit to the hospital
- the retention period for most of the on-line data is estimated to be in the range of ten years (to facilitate comparative annual, bi-annual, 5-yearly, and decade retrospective studies)

The system will potentially be used by anyone involved in the day-to-day operation of the Teaching Hospital (100-200 staff), by students enrolled in College programs (400-600), and by College faculty and administrators (100-150). Various external agencies and institutions are also interested in the exchange of and access to data, information and programs.

The method of collecting data from the various areas of the College is via an on-line system. This means that the computer is present in each operating unit, in the form of one or more display terminals. These terminals are used for data entry and retrieval “on the spot”; thus there will be little need to search through stacks of paper or wait for information to be written, transcribed, and transmitted from hand to hand. As an example, data entered by the Admitting receptionist in the Large Animal Clinic, and test results on the same animal entered on the terminal in the Clinical Pathology laboratory, will be “instantly” collated into one record, which is then immediately available to all hospital units via their display terminals. It is estimated that there will be a total of 70 TV-screens and 30 printing terminals installed in strategic locations throughout the College over the next several years.

Provision is made for editing and checking at the time of data entry, and for feedback editing to the individual generating the data. Some advantages of automating the gathering of data include: more rapid reporting, reduction in transcription errors, provision for centralized control of hospital functions, reduction in clerical workload for faculty, better scheduling of patient services, ability to capture patient charges as they occur, and greater ability to ensure compliance with policy.

The ability of the system to adapt to change (either of a technical nature, or in medical knowledge) is important. The files have the capability to accommodate the frequent addition of new types of information, without modification to existing programs. The system is able to retrieve data quickly either “horizontally” (by specific hospital/case number, owner/patient name, in order to assist with individual patient care) or “vertically” (by grouping cases by species, breed, sex, diagnosis, procedure, age, etc., for teaching and research purposes).

Privacy and the rights of individual owners and patients are of utmost importance. A security system is available which can be used to restrict access to groups of (i) users, (ii) patients, (iii) system features and functions, and/or (iv) specific fields of information.

Guidelines

For the following reasons, development guidelines, for use in an interactive environment, had to be defined:

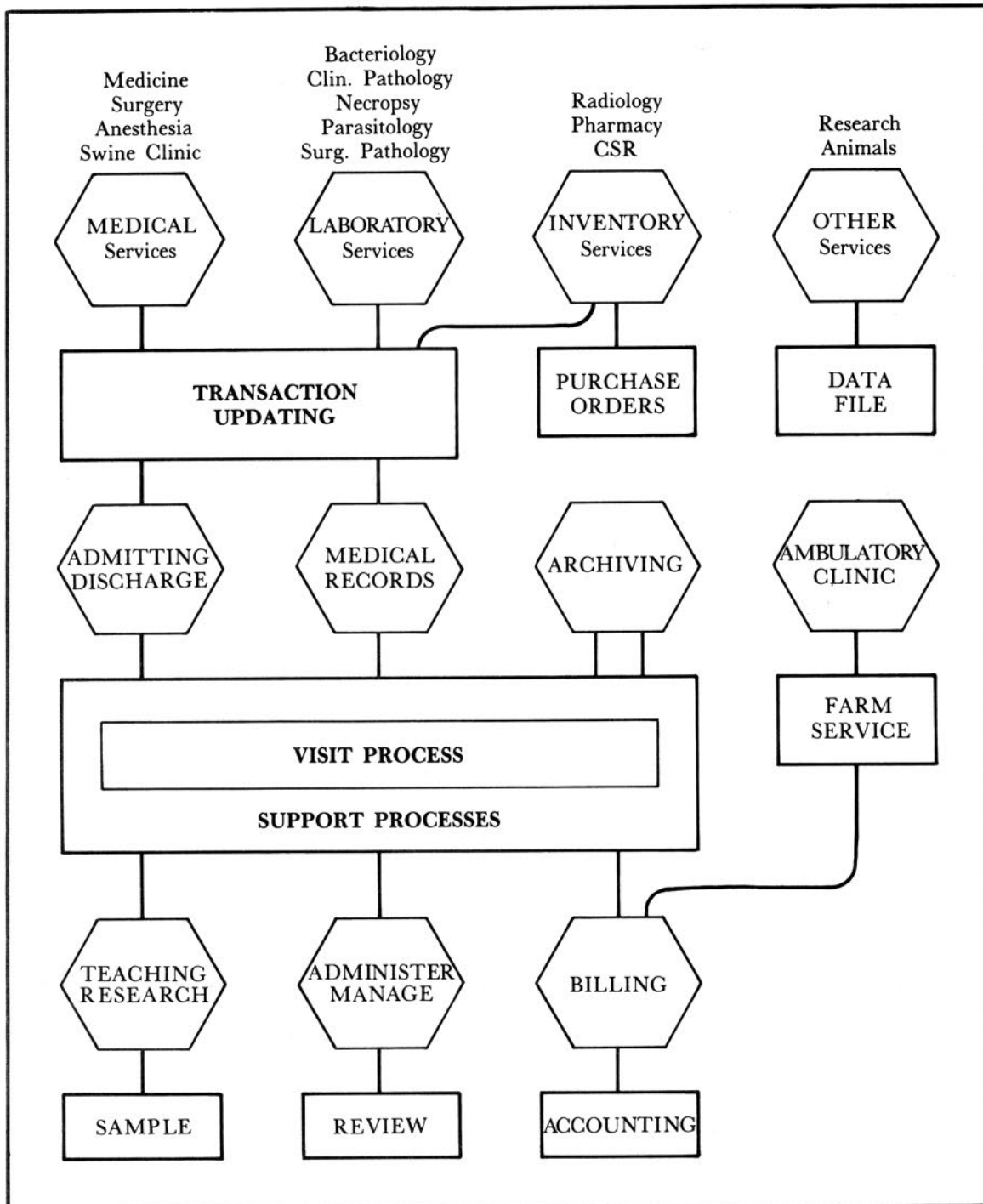
- 1) to maximize productivity of effort directed toward the development and maintenance of functional area projects
- 2) to provide mechanisms by which:
 - users and developers can communicate effectively
 - users can be involved in specifying, modifying, and evaluating the system during development
 - managers can monitor and participate in development

The guidelines include such items as:

- the project life cycle of each individual area, i.e., details of Phases V (Internal Design) through VIII (Review)
- within each phase, checklists of required activities to complete each stage, including the role of the user
- documentation requirements for manuals, e.g., packaging and content
- naming conventions for projects, groups, functions and variables
- function documentation requirements, and programming conventions
- “keyboard” conventions for user interaction in data entry and screen display

The above “standards” are necessary in order to maintain uniformity and consistency, both from a user’s “external” viewpoint and for a developer’s “internal” maintenance and support role.

Figure 1
External Design Context Diagram



Processes

Figure 1, a context flow diagram, shows the overall flow of information from the functional areas (in most cases) to the visit data base, and the interaction of these areas with the "support" processes. For reasons of both security and integrity, the support functions "protect" both the users and their data from accidental problems and/or malicious intrusions. They also afford the ability to provide the necessary audit trails (both medical and financial) to track activity both within case records and the Hospital.

Initial startup of the OVC system is identical for all users and is via a common APL public library workspace. Authorized users require a valid APL account number that is "registered" for use within the OVC system. Registration (via the User Services Assistant) defines for each individual user a personalized "profile" of those group(s) of functions that the user is allowed to use.

Users in the admitting areas of the Hospital are the only ones allowed to create (in "real time") the storage locations for new visits. All updating from every functional area is accomplished by submitting transaction update requests. In this manner, all users require READ ONLY access to files (at the APL system level).

One non-terminal routine (an N-task running under the auspices of Medical Records) is the only "user" in the system allowed to post information into any record. In addition, every transaction request, whether successful or not, is recorded into a monthly audit log. All information fields in the visit record are "shadowed" with "who" contributed the data and the audit transaction which yields the previous field value.

Admitting receptionists create (in the Visit file) the location (one file component) into which all information generated for the visit is kept. This process also assigns a unique "visit number", which is in fact the file component number. All additional patient and owner information (biographic, demographic, and signalment) that is provided is submitted and logged into the visit record via the transaction updating facility.

Billing clerks use recorded information in the visit record to calculate charges. However, until all areas of the hospital are "on-line", clerks will only be entering the charges levied (with the actual calculations being done manually).

Medical Records clerks enter "close out" information for the case, such as final diagnoses, procedures, etc., thus signaling that the particular visit is "inactive". The latter process is done only after the non-computer record is checked for completeness and has been properly filed in the medical records library.

Administrators, Managers, Teachers, and Researchers are able to "group" cases, for summarization and analysis, on a limited number of fields recorded in a "skeletal" or "quick-search" index. The latter index is implemented with the use of the I.P. Sharp product MABRA and is intended to be flexible so other search fields can be added as required. The index, because it contains frequently used search items (e.g., age, species, breed, sex, etc.), provides a fast method of searching the larger Visit file. The subfile produced can subsequently be searched "sequentially", or analysed as required.

Archiving of visit records is defined for completeness, but according to current definitions, is only required to be run annually in order to remove those cases which have had no activity for some yet-to-be-determined time period. The Archive Index is a cross-reference of those cases that have been copied to computer tape and hence removed from disk storage.

Visit Records are a composite repository for all source-oriented data generated during a visit. Each functional area defines, within the guidelines established by the Hospital administration, its own data requirements. Each visit record contains data only for those functional areas which have contributed to the case, and only for those fields of information supplied. For the benefit of the clinician responsible for the case, all fields are marked as to "who" supplied (i.e., entered) the information. This allows the clinician to verify and/or edit data provided by students. It is estimated that an "average" visit will generate about 15,000 bytes of data. The use of "enclosed" arrays has greatly simplified both the conceptualization and implementation of the above structure.

Names of owners, patients, clinicians, students, staff, referring veterinarians, truckers, contacts—i.e., *proper* names—are used extensively. For this reason, a names access facility is provided to allow all or part of a name to be entered. The system provides some limited qualifying information on any matches encountered to assist in making a choice. In addition, other "people" information is maintained strictly for the convenient and secure operation of the system. Addresses, telephone numbers, names of contacts, etc., are kept for customers, while for faculty and staff recorded items include their department, functional area, building/room location, etc.

Development cycle

An initial planning period, of about one month, by the Staff Analyst is designated to help the functional areas determine the quantity and type of data to be collected, as well as to identify the features required in their custom programs. This visit also acts as an orientation for everyone involved, and thus begins the user's education and training. At this time, required terminals and network connections are determined and ordered for delivery prior to any implementation of programs. This first visit also identifies the next stages (or "pieces") of development, and establishes their priority of installation. In most cases (in approximately the subsequent three months) the first "piece" is constructed by one of the Analyst Programmers. The same Analyst Programmer returns in 3-5 months to evaluate and to continue with the next stage in the evolution. In the likely event of problems, with any newly installed set of programs, the *original* programmer is "on-call" to resolve difficulties.

Conclusion

The increasing complexity of today's agricultural system necessitates that information be readily available to assist decision-making. It is hoped that the OVC Veterinary Medical Information Management System is a step in this direction, and that the information will be of benefit not only to the College itself but to others as well. The rapid advancements that have been, and are being, made in computer technology allow for the interfacing of data from other sources, e.g., farms, laboratories, etc. Some of the data generated will only be of benefit to those that generated it, while other information or summaries may be of greatest benefit when integrated with that from other sources. One of the many challenges for the future will be to design such overall systems such that they are cost-effective. The problem is not one of generating data, but one of generating decision-oriented data and of its interpretation.

ACTUARIES, MODELS AND APL

**Gary C. Mooney
Mooney Associates
Markham, Ontario**

Introduction

If you're interested in financial modelling, or if you've ever wondered what actuaries do, read on (actuaries included).

One definition of a mystique is "an atmosphere of mystery and veneration, associated with some creeds, doctrines, arts, skills, techniques or persons." It is fair to say that there is a mystique associated with each of actuaries, modelling and APL and, especially, with the combination of the three. This paper will try to dispel some of this mystique. As someone (possibly Tallulah Bankhead) once said, "There's less to this than meets the eye".

Comments, verbal or (preferably) written, on this paper are welcomed as it is intended to publish an updated version at a later date. Anyone who would like to receive a copy of the updated version is invited to request one.

What is an actuary?

Actuaries have been struggling with this question for many years and have not as yet arrived at a consensus. For purposes of this paper, we can use the following definition:

"An actuary is a person professionally trained in the evaluation and management of financial situations involving contingencies." (A contingency is an event that may or may not happen.)

In North America, actuaries qualify by completing a series of rigorous examinations over a period of several years after university. These examinations focus on mathematics and business theory and practice, and are sponsored by the Canadian Institute of Actuaries, the Casualty Actuarial Society and the Society of Actuaries. There are more than 10,000 fully qualified actuaries in North America.

Worldwide, requirements for qualification vary from country to country but, in countries having relatively large numbers of actuaries, are also rigorous. There are several thousand actuaries outside North America.

Some of the mystique associated with actuaries has resulted from their development, many years ago, of an extremely concise notation for use in describing actuarial techniques, and of ingenious methods for reducing the amount of manual calculation required to use these techniques. A significant portion of the training of actuaries has involved mastering the notation and learning the short-cut methods. Therefore, anyone without this training has been at a disadvantage when trying to understand or use these techniques.

The entry of actuaries into new fields and the increased complexity of today's world has outpaced the further development of actuarial notation. Computers have significantly reduced the need for short-cut methods. As a result, actuaries now tend to make more use of ad hoc notation that suits their particular circumstances and that resembles natural (spoken) language. Further, actuaries are now more inclined to use simpler methods in applying their techniques that recognize and exploit the power of computers. For non-actuaries, these changes mean that it is becoming easier to communicate with actuaries and to understand and use actuarial techniques.

What is a model?

A model is a representation of something in the real world. This paper deals with financial models. For our purposes, we define a financial model as a simplified representation of an enterprise involved in the provision of financial services and products, including insurance, deferral of income, loans and investments. This definition can include insurance companies, employee benefit plans, government insurance plans, banks and other organizations providing loans and/or investment products. With a little imagination, it can be extended to include the financial affairs of many businesses and of some individuals.

The paper focusses on financial models that are intended for planning purposes (outlook of more than one year) rather than those for budgeting purposes (up to one year). The approach described, in general, involves making specific assumptions about the future and testing their effects rather than using statistical techniques to define the future.

Financial modelling is based on accounting principles and results in the production of financial statements. The accounting profession has been working for some time to improve the usefulness of financial statements by refining accounting principles, encouraging standardization and developing new ways to present financial results in a more meaningful fashion. As a result, those interested in financial modelling are being provided with a better base for their models.

Actuaries and models

Actuaries have been heavily involved in financial modelling for more than a century and, as a result, are very comfortable with the concept of defining and using a model to represent events in the real world. The actuarial profession had its beginnings in the insurance industry and has expanded in more recent times into the employee benefits industry as well. These two industries are very much concerned with the financial implications of the future and use models to assist in planning for the future.

Originally, the valuation of liabilities related to future events and the design and pricing of products were of primary concern to actuaries. More recently, the valuation of assets, the relationship between assets and liabilities, cash flow and the emergence of earnings have become major considerations. Currently, actuaries are involved in every financial area of some insurance companies and of many employee benefit plans. Consequently, actuarial involvement in financial modelling is expanding to include all of these areas.

Actuaries, by virtue of their training, generally develop both a broad and in-depth knowledge of their particular enterprise and are thereby well equipped to model its operations.

Models and APL

For purposes of this paper, we shall describe APL as an algebraic notation that has been implemented as a general purpose programming language. Both of these facts are significant in financial modelling.

As an algebraic notation, APL provides a strong base for financial modelling. It is very powerful, meaning that one APL expression can describe or do a lot. It is highly consistent, an important feature as the model becomes more complex over time. It allows data to be described and manipulated in a collective manner, similar to the way in which people think. It is very well suited for defining the set of financial algorithms (defined later) which are used to describe the operation of the enterprise.

As a programming language, APL provides a practical implementation of the notation. It is application-oriented rather than system-oriented, meaning that both the developer and the user can focus on the business problem rather than on the underlying structure of the computer system. It is interactive in nature, providing immediate feedback (response) to both developer and user and, thereby, speeding up the learning process. The structure of the language encourages the use of a modular (building-block) approach in constructing systems, thereby facilitating implementation in stages and changes over time. It is very efficient in doing numeric computations, particularly in applications such as modelling which involve continual changes of data and algorithms.

As a base for financial modelling, APL offers a broad range of features that will meet most needs. Equally important, APL systems can be designed and implemented much faster and maintained with less effort than systems developed using other languages. Consequently, APL is being chosen by an increasing number of enterprises for financial modelling.

Actuaries and models and APL

Given the long-time involvement of actuaries in financial modelling and the usefulness of APL in this area, it is not surprising that actuaries were some of the first people to be attracted to APL and that growth of APL among actuaries continues to accelerate.

Actuaries have no difficulty in dealing with APL as a notation, having previously acquired experience in dealing with their own concise algebraic notation. They find the power of the language attractive as it enables them to solve problems and develop systems quickly and, in most cases, without reliance on non-actuarial programmers.

Myths

Only actuaries can do actuarial work. Not true. Actuarial techniques, some of which will be discussed in the next section, are not owned by actuaries. Anyone with knowledge of basic mathematics and basic accounting theory can add some actuarial techniques to his/her repertoire of skills.

Financial modelling requires sophisticated mathematics (particularly statistics) skills. Not true. Again, basic mathematics and basic accounting theory will get you well started and may, in fact, take you a long way.

APL is too expensive, unstructured, impossible to read, never documented, etc., etc. Not true. APL can be abused but, if used properly, can be extremely productive and cost effective.

Myths die hard and, therefore, it is expected that these myths will be with us for some time. However, let's think positively about what actuaries and/or actuarial techniques, financial modelling and APL can do for your enterprise.

Actuarial techniques

In a paper of this length, it is not possible to describe actuarial techniques in detail. What we'll do is describe the needs actuaries have identified over the years in doing financial modelling. In some cases, we'll describe techniques briefly, while in other cases we'll only imply the availability of techniques to meet the need. We refer to these techniques as "actuarial" not necessarily because they were invented by actuaries or because they are used exclusively by actuaries, but because actuaries have used them **in combination** in financial modelling with considerable success over the years.

1) Subdivision

It is important to divide the job of constructing a financial model into logical and manageable submodels. You can view the enterprise as a whole and start with either an income statement or a balance sheet orientation, producing the other (and other financial statements) as byproducts. Alternatively, you can view the enterprise in terms of the products it offers, producing financial statements for each and consolidating them into financial statements for the enterprise as a whole.

2) Data

There may be too much data available to allow modelling at reasonable cost or to enable you to identify the important characteristics of the data. You must find ways to summarize the data without reducing its relevance. On the other hand, there may be too little data or data that doesn't quite fit the circumstances. In this case, you must generate data by making assumptions or by modifying the best available data. Dealing with these situations is not particularly difficult if you have a good knowledge of the business.

3) **Algorithms**

An algorithm is a formula, rule or statement that describes some process. For our purposes, we'll define a financial algorithm as one that describes some aspect of a financial enterprise in a relatively simple way. "Pre-tax income equals revenue less expenditures" is a financial algorithm. You need to identify the important financial elements of the enterprise, then describe them in terms of a set of algorithms. You'll find that it's relatively easy to get started on this. However, as more refinement is desired and as dependencies are identified, you'll find yourself involved in increasing complexity.

4) **The time value of money**

A dollar tomorrow does not have the same value as a dollar today. If you're modelling a financial enterprise, application of the set of algorithms to the data will lead to the production of financial statements for future years. However, you shouldn't simply add up earnings or other items without regard to the years to which they apply. You must discount for interest to obtain the present value of earnings or, alternatively, accumulate with interest to produce the future value of earnings. You can use the same approach for other income statement items to obtain the present or future value of revenue from sales, investment income, sales expenses, payments to clients, administrative expenses, income taxes, and so on.

5) **Contingencies**

Repeating the definition given earlier, a contingency is an event that may or may not happen. There is a probability associated with its occurrence. Many people tend to think of only two possible values for the probability of occurrence of an event: 0 (it won't happen) or 1 (it's certain to happen). It is very useful and, in fact, often essential to consider the values between 0 and 1 as well. For instance, let's say that you know that something will happen within the next ten years, with an equal chance (i.e. 10%) of its happening each year. How do you reflect this in your model?

6) **Expectation**

If the event mentioned above will result in \$1000 of revenue, your expectation of revenue in each year is \$100 (10% of \$1000). If there are a sufficient number of such possible events, you can apply the probabilities associated to obtain the expectation of the amount of revenue in each year. You can also use this approach for other income statement items as well. If the events occur exactly as predicted, everything's fine. But in most cases, this won't happen. How do you deal with this situation?

7) **Variation**

There are a number of techniques to deal with random variation from expected. You can be somewhat conservative and build in a margin for results less favourable than expected. You can get slightly more sophisticated and look at several alternatives—for instance, best case, worst case and most likely case. You can assign a probability of occurrence to each of several alternatives. You can get more sophisticated still and use risk theory to give you some idea of the distribution of possible outcomes. But you should start with something simple and move to something more sophisticated only when you feel that it's necessary.

8) **Funds for stabilization**

One technique available to manage random variation involves the setting up of a fund or reserve to smooth the financial effects of random variation over time. When actual results are better than expected, the fund is increased according to some algorithm determined as being appropriate to the situation. When actual results are worse than expected, the fund is decreased in a related manner. Properly applied, you can use this technique to obtain a better understanding of the enterprise as a going concern by eliminating or reducing short-term distortions.

9) **Funds for matching**

Sometimes revenue is received in advance of the related expenditure. It is often inappropriate to apply this revenue to increase earnings in one accounting period and to reduce earnings in a succeeding period. Somehow you need to associate the two to avoid presentation of a distorted picture of future results. In some circumstances you can set up a fund or a reserve to accomplish this matching. You may also be faced with the reverse situation of an expenditure preceding revenue. In this case, you may be able to set up a negative fund or reserve.

10) **Pricing**

Let's define pricing as the process of determining what to charge for a product. With most financial products you have to consider factors that operate over time—the time value of money, expectation and variation—in setting a price that will meet profit and other objectives. One contingency that is often important in pricing is persistency, or the probability that a financial product will continue to be held by a client over a specified period of time. Persistency can have a very significant effect on both cash flow and earnings. Another factor that may require consideration is the cost associated with the temporary use of capital to support the product during its initial period. You must find a rational method of determining how to charge for your product that takes into account all of the important factors and that balances your needs with those of your client.

11) Projections

It is important to be able to estimate with some accuracy the effect of alternative courses of action on the financial enterprise. To do this you need the capability to project the financial results generated by your existing business, perhaps under several different scenarios. In addition, you have to be able to project results related to growth of the enterprise, and need the ability to consider several scenarios, likely involving alternatives relating to product mix, growth vs. earnings, etc. For both existing and new business, you need to be able to project these results a number of years into the future.

12) Analysis of results

Learning is greatly enhanced by feedback as to how you're doing. It is important to build a feedback mechanism into the financial model. Having chosen your course of action for some period of time, you'll have produced projections of expected results. You then need to be able to compare actual results with expected, after any appropriate transfers to or from funds for stabilization or matching. It will likely be necessary to make these comparisons at periodic intervals for major products over the lifetime of these products.

If you're an actuary you'll have recognized most or all of these needs as applying to your area of speciality. With a little thought, you may agree that the needs and the techniques to deal with these needs are similar for all specialties. Hopefully, you'll look for opportunities to apply these techniques to other financial situations as well.

If you're not an actuary, you may have identified one or more techniques that might be useful in your work. For more information, consult your local actuary.

How to do it

There are many ways to approach financial modelling. The approach you choose will depend on the resources available, the urgency attached to the project and the commitment of senior management.

In the following sections, we'll outline one approach that might be labelled simply as "start small and build gradually." This approach has been successful because it allows time for everyone involved, both technical staff and senior management, to acquire some experience in building and using models with minimal risk and disruption of current activities. It also permits development to proceed in parallel with management's changing perceptions of the enterprise's priorities.

Important elements

Some important elements involved in the construction of a financial model include: data, the set of financial algorithms, the control structure, documentation, and the user's environment. In this section we'll comment on the last three.

You need a control structure to provide you with the means to manage data and the set of algorithms, link submodels, update "gracefully", reproduce past results, analyze current results, compare alternatives for the future, produce reports and so on. Finan-

cial modelling should be viewed as being long-term and evolutionary in nature. Your organization must have a means of control over time that allows these activities to continue without significant disruption. In particular, it must be able to survive changes in both technical staff and senior management. Development of a proper control structure usually accounts for the major part of the work involved in constructing a financial model.

Proper documentation is vital to success in financial modelling. Without it, you cannot be sure that you have the model under control. In order to ensure that documentation gets done and to minimize the time required to do it, you should document the system as you develop it. If you leave the documentation until later, it will likely not get done.

You must place the user in a modelling environment—in effect, put him/her in the driver's seat. The environment must reflect the current implementation of the model and the circumstances of the user: computer system, APL product, terminals, hard copy devices, location of the user, his/her experience, any need to communicate with the non-APL world, and more.

How to get started

Prerequisites are: a good knowledge of the business, basic mathematics, basic accounting theory and, ideally, some experience in systems development.

Start with a need or needs expressed by senior management. Focus on getting something simple into operation quickly. Ignore the unimportant. Get some feedback from management and respond by making appropriate changes. Remember that you're doing this to assist them in their work.

Start with techniques that are simple and understandable to you and that can be explained to others. Document everything you're doing as you do it. Minimize your early commitment regarding computer equipment and staff. Assuming that you're using APL, one person using a timesharing system can accomplish a lot at reasonable cost.

As you progress, think more generally about the enterprise. For modelling purposes, try to identify logical subsystems. Start thinking about the design of an overall control structure. Pay increasing attention to the user's environment and the accessibility to and usefulness of the available data. Look for ways of improving the set of financial algorithms.

How to survive and prosper

Your success in financial modelling will depend heavily on how well you help senior management solve their problems—how well you've actually done it and how well management perceives that you've done it. In other words, communications with management will be of major importance.

You'll have to be prepared to make major changes in the structure of the system over time, as you gain experience, as the business changes, as management changes and as new technology becomes available. If you anticipate this need from the start, you should be able to plan for and make these changes with minimal disruption.

As you progress towards the complete system, you'll want to use more sophisticated techniques and pay more attention to dependencies among subsystems. You'll have to make sure that your technical knowledge keeps pace with the needs of the system.

Predictions

In general, some of the mystique associated with actuaries, financial modelling and APL will disappear. This will lead to increased use of these resources. But this will be a relatively slow process.

1) Actuaries

There will be an increase in the use of actuarial techniques by non-actuaries, assisted by actuaries and by financial modelling software. Actuaries will upgrade their technical capabilities and diversify into new areas. One area of increasing importance will be strategic planning, where actuaries will be able to combine their business knowledge and technical skills to great advantage. There will be more emphasis by the actuarial profession on development and use of computer skills. APL will become the primary actuarial programming language.

2) Financial modelling

This activity will become much more prevalent in financial enterprises. In fact, it will increasingly be regarded as an essential element by senior management in strategic planning and will contribute significantly to the financial stability and profitability of these enterprises. APL will become the most commonly used modelling language for financial enterprises.

3) APL

Predictions here are limited to those relating to the use of APL in financial modelling. APL will become generally available on microcomputers and will be useful in both standalone and distributed processing modes. The number of dialects of APL available will increase as will their disparity, with the result that more attention will have to be paid to portability of software. APL will become increasingly capable of interfacing with the non-APL world, making transfers of data easier. Enhancements will be introduced to improve APL's text management and graphics capabilities. APL will continue to develop to reflect advances both in technology and techniques and to meet the needs of the marketplace.

Conclusion

This paper has provided some information about actuaries, models and APL which, hopefully, will contribute to a reduction in the mystique associated with them. It has suggested that APL can provide an excellent base for the construction of a financial model. It has recommended the use of actuarial techniques to enhance the usefulness of the model.

Glossary

Definitions marked with an asterisk apply only in the context of this paper. All others apply more generally.

actuary	- person professionally trained in the evaluation and management of financial situations involving contingencies
algorithm	- formula, rule or statement that describes some process
algebra	- branch of mathematical analysis which reasons about quantity by the use of letters and generalized symbols
APL	- (1) algebraic notation invented by K.E. Iverson - (2) general purpose programming language
contingency	- event that may or may not happen
enterprise	- unit of economic organization or activity
financial	- pertaining to money
financial algorithm*	- algorithm that describes some aspect of a financial enterprise in a relatively simple way
financial model*	- simplified representation of an enterprise involved in the provision of financial services and products
model	- representation of something in the real world
notation	- set of symbols or characters used to denote things or relations in order to facilitate the recording or considering of them
persistence*	- probability that a financial product will continue to be held by a client over a specified period of time
probability	- ratio of the number of possible outcomes in favour of an event to the total number of possible outcomes
statistics	- branch of mathematical analysis which deals with the collection, analysis, interpretation and presentation of masses of numerical data
technique	- method of achieving one's purpose

MANAGEMENT ACCOUNTING IN INTERNATIONAL BANKING

G. B. Steinitz
Assistant Financial Controller
Midland Bank International
London, England

A management accounting system can be thought of as *the nervous system of an organisation*. Without such a system few businesses of any size would survive. The manufacturing industry and others have recognised this since the 1920's.

For a number of reasons many banks have not only survived but thrived without such systems and some still believe to this day that they can continue to thrive by flying by the seat of their pants.

Midland Bank International is not among those. We have recognised for some time the benefits of a sophisticated management accounting system. Before I describe it to you I think I had better tell you a bit about Midland Bank International.

We earn our money by our ability to attract funds below market rates and lend them above market rates—and to do so with sufficient margin to cover our operational and marketing costs, and the inevitable bad debts. We also perform a considerable number of money transfer and other services from which we earn commission. We operate a British Government backed export finance scheme. We buy and sell foreign currencies and indeed foreign notes. Broadly speaking that's it and it sounds relatively simple. In fact, accounting for this is rather complex, particularly when you consider that Midland is an international bank which is borrowing, lending, buying and selling anything up to 60 different currencies, whose relative values and interest rates are changing daily, is also entering into highly complicated syndicated loan arrangements, and is operating via branches, subsidiary companies, and correspondent banks in virtually every country in the world. Unlike a factory of comparable size—which would be mass producing or batch producing thousands, if not millions, of identical components and products—we produce millions of individual customer services. Each one is different; sometimes a little and sometimes a lot.

All earnings in foreign currencies are eventually converted to sterling. The value of those earnings changes daily with fluctuating rates of exchange. For example a hundred thousand French Francs earned in January is likely to have a different value from a hundred thousand French Francs earned in February when converted to sterling. What complicates it further is that the foreign currency profit reported in January, in sterling, will have changed its value by the end of February, and again each month after that until it is cashed in. In other words the profits reported for the month of January in sterling, plus those reported for the month of February, will not equal the cumulative profits for January and February, because the sterling equivalent of the

January foreign currency earnings will have altered by the end of February. Clearly, our management accounting system has had to be designed to take these and other complexities into account.

Our main commodity is money, and in my view there are few more perishable commodities than that. It can be confusing evaluating your earnings in the same terms as your sole commodity. It also has psychological drawbacks. A foreign manager used to negotiating loans of £50 million or £100 million at a time may find it difficult to appreciate why we are concerned about £10 thousand of expenses or a 2% drop in productivity. Despite these differences we have modeled our system on a structure very similar to those found in modern manufacturing companies, with cost centres and profit centres and managers accountable for achieving their budgets.

Profit centres' earnings are derived from the margins above market rates that they are able to make on loans, and the margin below market rates at which they are able to attract deposits. They earn commissions on the customer services and in turn are charged the operational cost of those services, just as a factory outlet would derive its income from its sales and be charged with the standard factory cost of the products sold.

Transfer pricing of funds borrowed or lent is done at the market rate applying at the time of each deposit or loan. Transfer prices of customer services are done at standard costs, with operational departments accountable for *recovering* their actual costs by the value of their output calculated at the standard cost per item.

Profit centres are accountable for profits much as you would expect to see in a manufacturing outlet situation.

Structure of the management accounts

We examined several ways of calculating and reporting the profits of the business.

The first, the conventional style of reporting profits, summarises the income by income category and expenses by expense category. This enables comparisons to be made between actual and budget, in a style to which people are already accustomed, and provides a quick and comprehensive view of the business as a whole. We call this the *financial report*. Its limitation is that whilst it shows where income arises and where costs are incurred it fails to show where *profits* arise. This is a major disadvantage.

The second looks at the business through the eyes of its managers. All income and all costs are attributed to the profit centres of the business using sophisticated processes of transfer price of money, product cost transfer, and overhead apportionment. We call this *responsibility reporting*. The advantage of this is that it introduces the concept of *accountability* for profits throughout the organisation. Its limitation is that it shows **where** profits are earned but fails to show **how**.

The third way looks at the profits of the business analysed by product group. This has tremendous advantages for product planning, marketing and subsequent control. We call this *product profitability* reporting. The limitation is that since the business is not structured by product groups, the figures cannot reflect the organisation, and lack the accountability reporting of the former.

Each of the reporting structures has advantages and limitations and we decided to

design a system which produced all three; that is, the financial reports providing a picture of the business as a whole, the responsibility reports, and the product profitability reports. Each of course ends up with the same amount of profits and, conceptually, it is perhaps best to think of the responsibility reporting and product group reporting in the form of a matrix, with the profit centres down one axis and the product groups along the other.

Customer profitability

As if three views of the business were not enough we have now introduced a fourth, namely *profits by customer*.

I think I am right in saying that banking is unique in that it is the only major business sector which deals in barter. It barter's customer services (either free or below cost) for the free use of part or all of customers' funds. This applies both to personal customer accounts and to working balances left by other banks. Consequently it is extremely important to evaluate and constantly keep under review the effect of this arrangement for each customer to ensure that each is, on the whole, a profitable arrangement.

Our customer profitability system produces a report for each customer, quarterly or on demand showing, *inter alia*:

- the average balances and their value to the bank in relation to the market value of funds in the currency in question
- the number of transactions analysed by type
- the cost of processing those transactions, the commissions earned, and the relationship between the two
- the profit (or loss) of that relationship for the period

It also evaluates, as far as we are able to do so, the value to us of the reciprocal work done for us by each correspondent banking customer.

This is essential information for the foreign managers if they wish to ensure that the business relationship with each customer is maintained at an appropriately profitable level.

Sub-systems

This, in broad outline, is the philosophy of our approach.

What I would like to do now is to describe to you some of the sub-systems which together form our integrated Management Accounting System. It is probably as well to remind you that not only is it integrated with the financial accounts, it is fully integrated within itself.

The significance of that is that any one looking at reports from any part of the system can feel confident that they are totally compatible with reports in every other part of the system.

Manpower and activity reporting system

We have a system of work-measurement which covers roughly half of the jobs in the Division. We are extending this to a further ten or fifteen percent, but the rest are unlikely to prove suitable for measurement.

This forms the basis of our Manpower and Activity report which is produced monthly for each department and shows:

- the actual number of staff
- the budgeted number of staff
- the budget flexed by actual volume of work
- the variance against the flexed budget
- the amount of absence compared with flexed budget and expressed in equivalent heads
- the amount of overtime compared with flexed budget and expressed in equivalent heads

In addition, for those departments whose work is measured:

- the volume of work expressed in standard hours, actual and budget
- the level of productivity

These reports are prepared from work-volumes and attendance figures supplied by each department monthly. The system evaluates the work volumes into standard hours and produces the report. This is used for controlling staff numbers and is an essential means of cost control in an organization where more than half of the costs are staff costs and a further 15% are staff related.

It is also an integral part of the Expenditure Reporting System.

Expenditure and variance analysis reports

The expenditure and variance analysis reports are produced monthly, showing for each department actual and budgeted spend for the month, the past three months, and the year to date in about a dozen cost categories. The measured departments have a comprehensive variance analysis showing the spend not only against the original budget but against the costs recovered by actual output. Variances are broken down by:

- efficiency
- volume
- pay rates
- spend

This enables the manager of each department to see precisely the effect on profits of the activities of his department *in relation to actual volumes* of output. We achieve this by comparing actual costs both with budget and with *cost recovered* by the volume of products actually produced: a notion quite common in manufacturing industry but still somewhat new in banking.

The reports are supported by two further schedules detailing precisely how his costs have been made up—down to individual items and including line by line the effect of expense accruals. These departmental expenditure reports are summarised and applied to relevant supervisory management levels and are used as the basis for a monthly examination of any deviations from budget.

In addition, we produce reports which analyse the expenses of the business as a whole, by type of expense, and have the facility to call off a report under any expense heading showing the breakdown of that expense category by department, comparing actual with budget.

We also produce some specialised reports on things such as space costs where the nature of the expenses is unique and requires different treatment. Space costs in London are very high and we are further developing our system to evaluate better the cost-effect of space utilisation. In the final analysis what counts in controlling space costs is **the cost of space per person**. This will vary with the amount of space he actually occupies, the amount of unutilised space in the building where he works, the costs per square foot of each element of cost (rent, rates, heating, lighting, security, and so on). Our system will provide a comparison of the space cost per person between the different locations, supported by a detailed analysis of the reasons for differences in costs.

Our responsibility reporting system, and indeed our product cost system, requires us to make an accurate calculation of the impact of overheads on all departments. Consequently we have developed a sophisticated system of overhead apportionment which is based on budgeted expenditure levels. Our calculations are based on conventional cost accounting practice of the utilisation of each overhead activity by user departments. It becomes complicated because many departments are inter-dependent and costs literally go round in circles. Having specified the patterns and the formulae, the system uses what is termed a “squirrel cage” technique to circulate these costs until they are all but eliminated.

Product cost system

The concept that a bank produces *products* is comparatively new to us but, in truth, a bank is not very different from any other business. Each loan and each deposit is a product. So is each Documentary Credit, Inward or Outward Bill, Cheque Paid and so on. We have identified 10 main product groups. The most complicated is “Payments Transmission” which includes all the payment instruments such as those I have just mentioned and many more. In fact our product cost system evaluates 124 different types of transactions. The calculations are based upon the standard times calculated for each work-unit by our work measurement system, and the individual departmental budgets of work volumes and costs, including overheads.

As I mentioned earlier, a bank does not produce batches of identical components or products but rather very large volumes of individual customer services. Our method of calculating product costs has to take this into account, and the only possible approach

is to calculate statistically valid averages. Our system has defined the “components” of each product and the departments where they are processed. Each year we analyse the previous year’s activities in some considerable detail and calculate the average number of times that each component arises for each average product. This provides us with our file of *product structures* and shows for each product the list of components, the number of times that each component occurs in the product (frequently a fraction of 1) and the number of standard minutes required to produce that component.

I will describe our budgeting system shortly, but it is from this system that we obtain the total budgeted costs of each department, the total output in standard minutes and, thus, the cost per standard minute.

The product structure file takes the standard minute cost values of each department, applies them to the relevant components, and evaluates the *standard cost* of each product.

The standard product costs are recalculated annually and take into account changes in method and average component mix—and of course newly budgeted levels of productivity and cost. As I will explain shortly, this system and the budgeting system are closely inter-related.

The *product cost system* is a fundamental part of our whole management accounting system and forms the basis of both the responsibility reporting system and the product profitability reporting system. It incidentally also plays a very useful role in the matter of cost control. Each product structure is subjected to rigorous scrutiny to identify areas for possible cost reduction.

In the final analysis what counts is the ability to achieve optimum (low) unit costs. To understand the meaning of this is to understand the meaning of *cost effectiveness*.

Budgeting system

Our financial year ends on the 31st of December and our budgeting season begins in September.

The budget system is a detailed and systematic process which begins with a detailed forecast of the results for the remainder of the current year.

First, we prepare a broad-brush “top-down” budget to enable the Policy Committee to establish budget objectives. Following this, each profit centre provides details of business volumes and the resources that they require to produce the business. Our Economics Department provides us with the economic assumptions of interest rates and exchange rates, and our Personnel Department provides us with the pay rate assumptions.

The inter-relationship between the different aspects of the budget makes the processing of the data quite complicated. For example, volumes of budgeted payment transmission transactions are supplied by the profit centres. These are converted via the product structures into standard hours analysed by operating department. The departments have, in the meantime, prepared budgets of average attendance hours per person per year, absence, overtime, and productivity.

Combining these makes it possible to convert the standard hours into attendance hours and then into the average number of people required to complete the work. The operational departments then have to break this down into actual people by grade, phased over the months of the year. This then gets further processed, together with the pay and other staff cost assumptions, and provides us with the manpower and staff cost budgets.

There is a similarly systematic process for the other aspects of the budget. The consequence is a very detailed budget report, profit centre by profit centre and cost centre by cost centre, phased over the months of the budget year. This goes through the normal review and adjustment process after which the overhead apportionment process takes place.

This budget then forms the basis of the calculation of standard product costs explained earlier, and of course forms the basis of our reporting and control system.

Responsibility reporting

Each manager receives a report each month, with details of his performance against budget for the month and year to date. We produce a profit and loss account for each profit centre showing their income and expenditure broken down by product group, with full back-up details. As I mentioned earlier, the performance of the operational departments is measured not simply by comparing actual expenditure with budgets but with the costs *recovered* by the volume of the products actually produced. It is this *recovery* which forms the charge to the profit centres for those products sold. This requires a continuing analysis of products by profit centre and indeed by customer within each profit centre.

On the whole, overhead departments' costs are recovered at the budgeted level. This recovery is charged via the overhead apportionment system to the user departments and eventually must *all* find its way either directly, or via the products, to the profit centres. All the income naturally finds its way to the profit centres and, in the way just described, all costs do too; that is, direct costs, product costs and overheads. Thus the profit centre reports reflect as far as is possible the true profits attributable to them.

In addition, we supply a complete set of summary reports of the business as a whole to each General Manager and Assistant General Manager, a circulation of about 35. It is a hierarchical set of reports and the intention is that people use it as a sort of telephone directory, alerted by the variance analysis summary or other high level reports at the front of the package to points of interest detailed in subsequent schedules.

This in very brief outline is our management accounting system.

How does APL fit into this and where do we go from here?

To be perfectly honest, we came across APL by accident. Our existing management accounting system was showing its many limitations some three years ago in several areas. The area in most urgent need of attention was the International Treasury. The Treasury is responsible *inter alia* for taking and managing all the currency deposits required for purposes of the commercial lending done elsewhere in the bank. Once in the market for taking deposits it means they are also in the market for making interbank loans. So their profits arise from their ability to take deposits below market rates plus the margins they can obtain from lending in the money market.

In addition they are responsible for managing retail currency deposits—both interest bearing and call accounts. They needed to know, far better than we had been able to tell them until then, exactly how and where their profits had been earned each month.

We developed a system which provided the necessary information but it was obvious that this could not be operated manually. I.P. Sharp were being fairly extensively used by Treasury at that time for operational systems, and we borrowed one of their programmers to implement a stand-alone Treasury management accounting system for us, using time-sharing.

In just 4 weeks we had a program working which produced detailed meaningful reports, available in eight currencies, converted to sterling at budgeted and actual rates of exchange, and supported by a detailed variance analysis. I was greatly impressed with the speed and flexibility of APL and set about developing and implementing the sophisticated integrated management accounting system I have spoken about today.

The advantage has been that it has enabled us to get a fair proportion of the system up and running with great speed. Its flexibility has enabled us to develop and often change our ideas as we went along. It was of course obvious to us that with volumes of transactions as high as ours, particularly on the customer profitability system, we would soon reach a point where time-sharing would no longer be the most economical solution in the longer term, and we are now redesigning and enhancing the system to be operated on our own in-house computer.

At this time, we are still totally dependent on time-sharing and for us it has proved to be a most effective and profitable means to a very important end.

CURRENT DEVELOPMENTS IN SHARP APL

Eric Iverson
Data Processing Manager
I.P. Sharp Associates Limited
Toronto, Ontario

Introduction

The fourteen years of continuous development of the collection of software products that constitute the SHARP APL SYSTEM have seen a gradual shift in emphasis through three major phases. The first two phases have been thoroughly documented in previous conference proceedings [Refs. 1, 2], and will only be summarized briefly here; the third and current phase will be discussed at greater length, particularly with respect to the *combined result* of the developments in all three phases.

The first phase, started in 1969 when we installed IBM's APL\360 in our then fledgling data centre, involved the design and implementation of extra-lingual facilities necessary to make APL viable and competitive in the world of large, business data-processing applications, and included eight key developments:

- 1) File System—This set of quad functions allows convenient, efficient, shared access to arbitrarily large and arbitrarily organized data bases stored as APL files.
- 2) `□FMT`—This function allows convenient and efficient report formatting.
- 3) `HSPRINT`—This utility formats and prints APL files on high-speed printers.
- 4) N-tasks and B-tasks—The ability to have APL tasks (active workspaces) run without terminals and to be automatically scheduled allows non-interactive, "batch" type applications to be designed and implemented in APL.
- 5) PACKAGES—This data type and the associated quad functions allow for convenient and efficient manipulation of collections of functions and variables. Together with the file system they provide necessary tools for dealing with large, complex applications.
- 6) Crash Recovery—This facility provides *CONTINUE* workspaces even in the event of total system failure.

- 7) Event Trapping—This set of quad functions and variables allows program control over a large range of errors and other events.
- 8) Large System Capability—Considerable development effort in the area of schedulers and resource managers allows the support of large numbers of concurrent users.

The second phase, started in 1976 with our decision to make the SHARP APL SYSTEM available as a software product to run on in-house computers, involved development in two key areas: the integration of SHARP APL into MVS, the mainstream IBM production operating system, as a proper subsystem, and “packaging” the product so that it could be installed, maintained, and updated at a large number of sites around the world. Currently we have over 30 installations, with two in the Far East, five in Europe and Africa, and the remainder in North America.

We are now well into the third phase of development, which concentrates on designing and implementing *interfaces* giving the end user, through APL applications, convenient, efficient, and secure access to *all* aspects of the data-processing complex. Details of some of these interfaces are presented in subsequent sections.

In summary, the first phase made SHARP APL a strong, self-sufficient and complete system; the second phase allowed that system to run in MVS, and changed a single-site, service bureau offering into a software package that could be installed in any corporate data centre; and now the third phase provides interfaces between APL tasks and all other activity and data in the data centre. The combined result of these developments is an APL system which can easily become the cornerstone of any serious attempt at implementing an effective *information centre*.

Information centre

Periodically the data processing community is faced with the use, misuse, and overuse of a new phrase. Having recovered from *distributed processing* and *personal computing*, we find that the phrase of the 80's (or at least the early 80's) is *information centre*. As is often the case with a phrase that gains quick and widespread use, there is a very valuable insight buried under a lot of rubbish. Usually the insight is something that has been around for quite a while, but has in time simply gained enough acceptance and understanding to suddenly be widely popularized. This is particularly true of the phrase *information centre* which summarizes a most important concept—that is, putting the end user in control of his own data processing.

Hammond's paper “Management Considerations for an Information Center” in the *IBM Systems Journal* [Ref. 3] is an informative presentation of the concept and its implications, and the following quote from that paper is a good description of an information centre:

The objective of an I/C is to provide users access to data on their own terms so that they can solve their own business problems. It is typically accomplished by providing a set of packaged tools and data availability (with appropriate training and consulting support) to the users enabling them to gain the power of the computer in a relatively easy and timely fashion.

To people familiar with APL time-sharing this is a concept that goes back to the

invention of the language; now, more than a decade later, we in APL find out that all along we have been in the information centre business. The interesting thing is that to the non-APL data processing community (still quite large, unfortunately) the phrase represents a new concept.

Hammond discusses the importance of applications written in APL in the operation of an information centre, but only as discrete, isolated components. The importance of tying *everything* together is implicit in the concept, but APL's potential in this area is ignored; probably due to IBM's treatment of APL as just another language processor, to deficiencies in their various versions, and in particular to the lack of appropriate interfaces to the rest of the environment.

Interfaces

Five general categories of interfaces will be discussed; all are between an APL task (active workspace) and some other, usually non-APL, part of the environment. All interfaces are between asynchronous processes and are implemented by the use of Shared Variables. The asynchronous process which implements an interface is called, in shared variable terminology, an Auxiliary Processor (AP) and each AP is identified by both an interface name and an AP number (for example, the STSK interface is AP1).

In all but one case the shared variables are directly visible to, and are manipulated by, functions in the workspace. In the STSK interface the shared variable is managed directly by the system and is not visible as such to the workspace. All the interfaces have been designed with careful attention to function, convenience, efficiency, reliability, and security. All of these, particularly security, are extremely important, but will not be treated in detail or belaboured in the following discussions.

A good, general discussion of shared variables can be found in the *SHARP APL Reference Manual* [Ref. 4].

Terminal interfaces

Logically all terminals are connected to an APL task through the STSK interface [Ref. 5]. This interface allows an AP to provide input to an APL task, to process output from an APL task, and to manage the state of an APL task; all completely device independent. In many cases the AP may map the APL input, output, and state on to a real device. In other cases there may be no real terminal involved at all.

Any processor with access to shared variables can use the STSK interface to sign-on to APL and to run APL applications. For example, an APL task can share a variable with the STSK processor and can initiate and manage another APL task. Similarly a program written in assembler, COBOL, PL/1, or any other language and run in batch (or under TSO, etc.) could easily initiate an APL task and provide input and process output.

Three auxiliary processors that support real terminals are provided as part of the SHARP APL SYSTEM.

The first, MPXH (Multiplexor Handler), is the oldest APL terminal handler. It

supports asynchronous terminals through the byte multiplexor subchannels of the CPU and an appropriate front-end processor, which can be any of the following:

- 1) An IBM 3705 running the SHARP APL NETWORK SOFTWARE; supporting asynchronous terminals *and* an interconnection to the I.P. Sharp worldwide network.
- 2) An IBM 3705 running the IBM Emulator Program; either by itself, or as part of a Network Control Program.
- 3) An IBM 270X.
- 4) Any hardware or hardware/software equivalent to the above.

The second, IDSH (Information Display Station Handler), supports the IBM 327X family of display stations when connected through VTAM (IBM's Virtual Terminal Access Method). IDSH supports these terminals through four interfaces:

- 1) The Session Manager uses the STSK interface and supports the terminal as a sophisticated, intelligent terminal with many useful capabilities. It maintains a file of the session so that the user is not limited to the last screen image, but instead can browse through his session; it supports APL and various non-APL keyboards and character sets; and it provides user and program control of Program Function Keys and extended features (for example, colour and highlight coding of different fields and types of input and output).
- 2) The ARBIN Manager uses `□ARBIN` and `□ARBOUT` and gives the programmer direct access to *all* the hardware capabilities of the terminal. It is useful for sophisticated applications that use hardware features not accessible through other interfaces, and is particularly useful for experimentation with new hardware facilities.
- 3) AP124 is an AP that is part of IDSH, and is an enhanced version of IBM's AP124. It provides convenient and efficient alphanumeric fullscreen support for the terminal.
- 4) AP126 is an AP that is part of IDSH, and is equivalent to IBM's AP126. It accesses IBM's GDDM (Graphic Data Display Manager) and provides alphanumeric and colour graphic fullscreen support for the terminal.

The third terminal handler, NTOH (Network Terminal Option Handler), supports the IBM 3767 terminal when connected through VTAM, as well as asynchronous terminals connected through NTO (IBM's Network Terminal Option).

The design of the STSK interface, along with some powerful development tools and techniques (prototyping in APL), makes it very easy to develop new terminal handlers. This is particularly true for any device supported through VTAM. For example, NTOH was designed and implemented in less than two months.

File interfaces

At a certain level all data is stored as an Operating System Data Set file, as a VSAM file (IBM's Virtual Storage Access Method), or as an APL file. In some cases the data is not usefully accessed directly at that level, but rather is accessed through a "data base access method", such as IMS or ADABAS, which uses a more complex structure built on the underlying files. However, in many cases the data can be usefully accessed at the file level.

SHARP APL can now directly access data in all three forms of files. APL files are accessed by the file system quad functions, VSAM files are accessed by AP123 (an enhanced implementation of IBM's AP123), and Data Set files are accessed by AP370 (an enhanced implementation of IBM's TSIO). AP123 and AP370 access files in the same way as any non-APL program would access those same files, giving an APL task the same access as any other task. Similarly, non-APL tasks can access APL files directly, just as an APL task would access them.

A key point is that there is no need for data to be "extracted" or to be stored in multiple places in multiple ways. Rather, the data can be stored in one form, and regardless of the form, it can be accessed from APL.

VTAM Interfaces

VTAM (IBM's Virtual Terminal Access Method), a general facility for communicating between asynchronous processors, is analogous to the SVP (Shared Variable Processor) in APL. In most cases one processor is an application system such as TSO, and the other is a real terminal, but in general the terminal could be any program with VTAM access. The application system interface accessible through VTAM is analogous to the STSK interface to APL accessible through the SVP.

The AVAM (APL VTAM Access Method) auxiliary processor provides the interconnection between these two communication facilities—VTAM and SVP. Hence, AVAM allows communication between any task which can use the SVP and any task which can use VTAM. Increasingly this means that just about any task can talk to any other task.

The most common use of AVAM involves APL tasks accessing VTAM applications. TSO, IMS, CICS, and ADABAS are a few of the many applications accessible to APL in this way. In addition to programs, real devices, such as an IBM 327X hardcopy printer, are also accessible to APL.

Through AVAM, with a few simple functions, an APL task can sign-on to a VTAM application and provide input and process output. For example, an APL task, entirely under program control, could sign-on to IMS and process IMS inquiries and updates in combination with data from other sources. It is important to note that, because a normal user interface to the VTAM application is used, all security requirements are strictly observed. The APL task connects to the VTAM application exactly as the user would with a real terminal, with the significant difference that the processing is all under the control of an APL application, with all the flexibility and power that this implies.

AVAM interfaces to data base management systems like IMS are particularly important because they complete the requirement to be able to access *all* data from APL.

The File Interfaces provide access to the low level files, and AVAM provides access to the higher level "data base access methods".

The fact that VTAM has network capabilities is also significant. One of the early uses of AVAM was to access IMS data bases on a machine thousands of miles from the machine where the APL task was running. The distance was transparent to both IMS and APL.

AVAM can be used in many different ways. It can be used to provide essentially a transparent "pass through" facility. For example, a real terminal supported by APL, but not supported by a VTAM application, could be supported through a simple APL application which mapped the unsupported terminal onto a type that was supported. In this case the user would not be aware of APL at all, and would only see the VTAM application. A more common use would be an APL application which accessed a VTAM application that was completely transparent to the user. For example, a user inquiry to an APL application for a report that required data from APL files, VSAM files, and an IMS data base could have all the data collected, analyzed, and reported without the user being aware of the details of the exact source of the data.

Non-APL function interfaces

FCAP (Function Call Auxiliary Processor) provides a very general interface from an APL task to any function, written in any language, available in the environment. The APL task accesses such a function by setting a variable shared with FCAP with an enclosed array containing the name of the function and the values of the arguments. FCAP does any required data conversion, calls the function, and sets the variable with the results, again with data conversion as required.

Initially we expect FCAP to be used primarily to access existing routines, largely written in FORTRAN, in libraries of mathematical and statistical functions. Eventually we expect to see a number of custom designed functions, satisfying unique requirements at a particular site. It is important to note that, although the main intent is to provide access to "functions" in the formal sense of routines with arguments, results, and no side effects, there is in fact no such restriction, and any routine that can be called can be accessed. In fact, the first routines made available to FCAP in its beta-test install were the batch access routines for ADABAS, thus giving APL tasks access to ADABAS.

MVS interfaces

IMVS (Interface to MVS) provides an interface to the MVS Job Entry Subsystem (JES), which allows the submission of batch jobs by an APL task and allows batch job output (SYSOUT) to be placed in a designated APL file. IMVS uses the JES Internal Reader and Internal Writer facilities that are analogous to the STSK interface to APL.

An APL user task does not access IMVS directly to submit a job, but rather communicates its request via shared variables and files to a system N-task that alone has authority to access IMVS. This centrally maintained N-task can enforce any required rules on submitted jobs (for example, job card and accounting information). A more sophisticated N-task processor called JCP (JCL Control Processor) takes user requests

in the form of high-level specifications and automatically builds the appropriate job to pass to IMVS.

Similarly, SYSOUT directed to APL passes through a system N-task that determines where to place the output and what data conversions to apply.

An additional MVS interface, provided by TSIO (AP370) to authorized system programmers, allows APL tasks access to MVS system consoles. MVS console commands can be entered into the system and MVS console screen images are available for display or analysis. This is a considerable convenience to a system programmer whose terminal can now, for example, switch between his APL application and being an MVS console at the touch of a key, but is of little interest to the normal end user.

All together now

With appropriate use of the right interfaces, convenient, efficient, and secure access can be provided from an end user APL task to any data or capability in the environment. This ability to access and manage *all* of the data centre from APL makes it possible to present a single, clean, and consistent image to the end user, regardless of his current or future requirements.

Most importantly, with the user's view of the data centre implemented in APL, that view can naturally expand in scope and in detail as the user becomes more experienced and sophisticated. The neophyte user can have an environment that is completely under the control of an application system set up by the professional staff of the information centre. Typically such an environment would be restricted to a set of particularly well documented and supported applications that were user-friendly, menu-driven, and with considerable programmed "help" available. As the user gains experience and expertise, and as he starts to pose problems not within the scope of the initial environment, he can easily be allowed access to lower, more general levels of the original applications. Similarly he can be allowed access to other applications that weren't appropriate for a naive user, and eventually will start to build his own applications out of the building blocks provided by the information centre.

All the interfaces discussed, plus others, are complete and proven parts of the SHARP APL SYSTEM. As interfaces they are basic tools required to solve problems; in some cases they will be used directly by the end user, but usually they are used in combination with APL applications by the professional staff of the information centre to build end user applications. Using these building blocks, many of our in-house customers have taken quick and impressive steps in setting up highly-valued information centres.

Nevertheless, considerable work remains to be done in continuing and expanding the role of SHARP APL in an information centre. The basic interfaces and some significant applications (such as MAILBOX, MAGIC, MABRA, CONSOL, and SUPERPLOT) are there, but work needs to be done in providing new applications, particularly applications which use, and facilitate the use of, the interfaces. The fact that the system level work is largely done is encouraging, as it means that the remaining work can all be done in APL.

The future

Development will probably never be finished in any of the three phases discussed. There will always be projects like performance improvements and new APL file quad functions (Phase 1), integrating APL into the new MVS/XA operating system (Phase 2), and interfacing to a mass-storage subsystem or to a hardware array processor (Phase 3). A phase, once started, does not come to an end. Development effort will always be spent on all phases, but the emphasis will shift. Currently most effort is spent on the third phase of interfaces and this will continue for a while, but the emphasis is already shifting to a new, fourth phase.

The fourth phase involves the design and implementation of changes and extensions to the APL language itself. The design stage of the fourth phase, witness the published literature, has always been quite active, but implementations have seen remarkably little change to the language proper between 1969 and 1980, with the notable exception of shared variables introduced in APLSV in 1973.

The fourth phase started slowly in 1980 with the introduction of extensions to the domains of *compression*, *and*, *or*, and *grade*. Since then it has continued with the introduction of complex numbers, general arrays, *enclose*, *disclose*, and several new operators. The coming years are going to see considerable development in this area and I look forward to reporting on it in detail at the 1984 APL Users Meeting.

Summary

SHARP APL has evolved extensively over the last fourteen years and, if anything, the rate of change will increase in the future. It is now a high-performance, high-function timesharing service available both on commercial service bureaus, and on private in-house systems. Either through a service bureau (particularly with a large number of public data bases) or in an in-house data centre, or in combination, SHARP APL can be the cornerstone (or indeed the whole building) of an information centre.

References

1. Iverson, E.B. "The Commercializing of APL\360 and Some Plans for Its Future." *Proceedings of the 1978 APL Users Meeting*.
2. Iverson, E.B. "The Integration of APL into the Larger World of Data Processing." *Proceedings of the 1980 APL Users Meeting*.
3. Hammond, L.W. "Management Considerations for an Information Centre." *IBM Systems Journal*, Vol. 21, No. 2, 1982.
4. Berry, P.C. *SHARP APL Reference Manual*. I.P. Sharp Associates, 1979.
5. Lathwell, R.H. "SHARP APL Multiprocessing and Shared Variables". *Proceedings of the 1980 APL Users Meeting*.

THE PROMIS MINE PLANNING AND CONTROL SYSTEM

**Walter John Corrie
Project Manager
Mining and Production Services
INFOGOLD
Orange Free State, South Africa**

INFOGOLD is a computing centre which serves the nine Anglo American Corporation gold mines in South Africa. It has two large IBM machines and supports a network of some 500 IBM 3270-type screens. The majority of systems are COBOL/IMS with a growing number in SHARP APL.

This paper describes a production planning and control system. It also relates the process of learning by discovery which has occurred over the four and a half years since APL was first introduced.

The mine planning and control problem

The gold bearing reef in South African mines is a sloping plane, usually less than one foot thick. To extract it a layer about four feet thick is removed by drilling and blasting. This activity is called stoping.

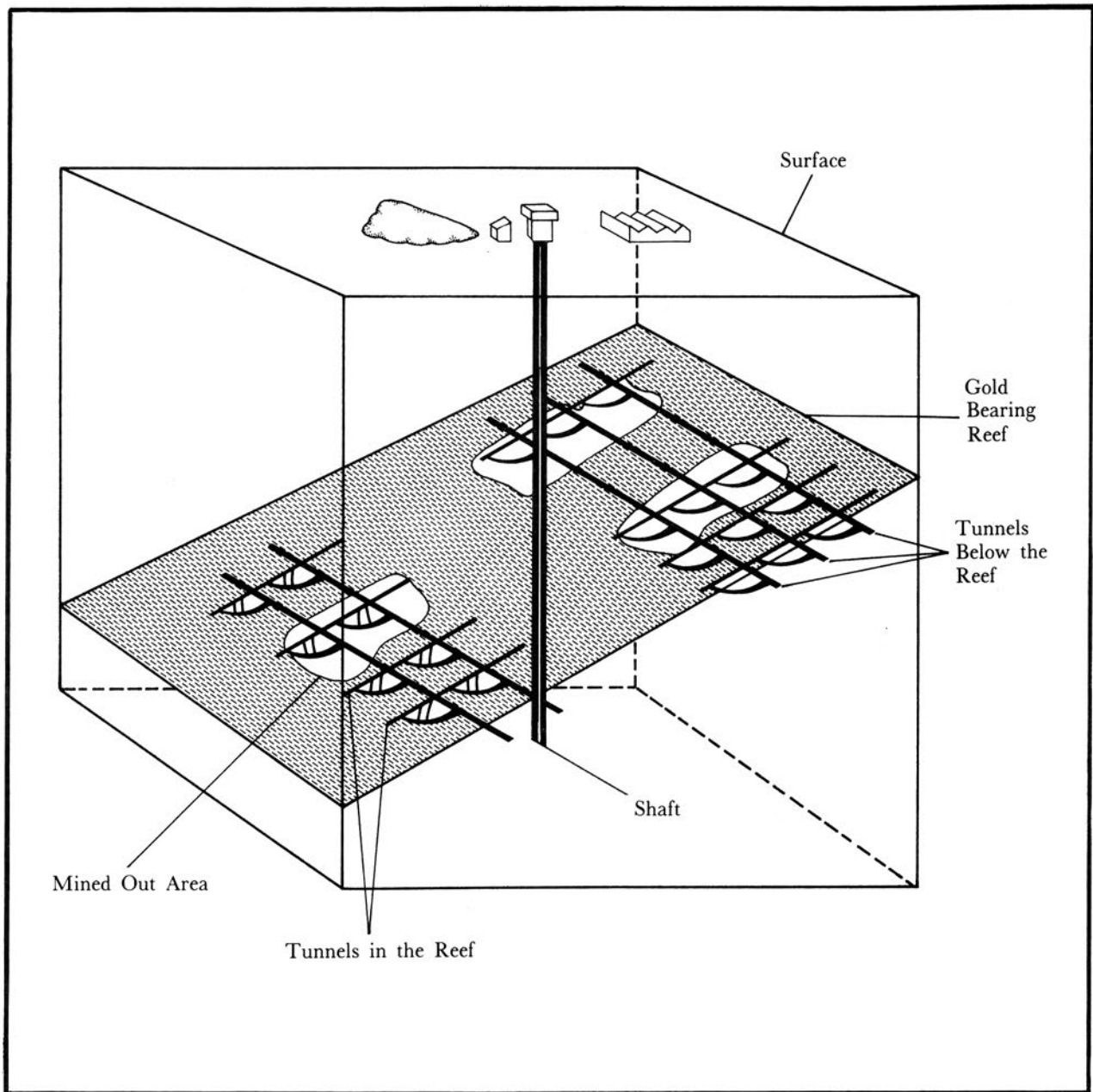
The space mined out closes up after a period of three months to two years, depending on depth. Access to the stoping areas is via a network of tunnels below the reef (Figure 1). Some of these tunnels are equipped with rails for small locomotives.

Too much development of tunnels ahead of stoping ties up capital, but provides the opportunity to seek appropriate gold grades in times of changing gold price, or when mining problems are encountered. Too little has the opposite effect and could lead to a situation where ore tonnages drop due to the shortage of available face. The need to co-ordinate the activities of teams developing new tunnels and those extracting ore creates a scheduling problem.

Typically the shaft planners produce a 24-month schedule of work for the planned tunnels and stoping panels each year, and update the first 6 months of it every 3 months. Having developed a schedule, it must be evaluated by calculating the ore and waste tonnages, quantity of gold and uranium, etc.

Figure 1

The Gold Reef and Network of Tunnels for Access



This is an onerous process when done by hand, particularly when the Manager rejects your first three attempts because the totals don't look right. The temptation to change the totals without the detail increases with each iteration.

Every month the managers of the mining sections within a shaft produce detailed one-month plans. At the end of the month actual progress is measured by the Survey department and the mine layout drawings are updated.

Here then we have the information required to produce variance reports against the one-month and longer-term plans.

Description of the PROMIS system

The planning system allows the development of a schedule of work on a screen (3270-type). It evaluates the schedule by calculating tonnages, gold, uranium, etc. It also provides for the entry of monthly plans and actual results, and produces variance reports.

The following describes the process in more detail.

Figure 2
Scheduling Screen

WP NO	WP NAME	TOT	OCT	NOV	DEC	JAN	FEB	MAR	APR	MAY	1980	JUL	AUG	SEPT
7	57/20 CROSS CUT	41	14	14	13									
8	57/20 HAULAGE	72	24	24	24									
9	57/20 BOX 9	40	10	10	20									
10	57/21 RSE EXT 3	35	20	15										
11	57/21 WZE EXT 2	92	20	24	14	34								
12	57/21 WZE EXT 1	180	10	20	10	20	20	20	20	20	10	10	10	10
13	57/21 BOX 4	70					10	10	10	10	12	11	9	
14	57/22 I/L BOX	25			25									
15	57/22 HENN DR. S	24				24								
16	.	34				24	10							
17	.	40					20	16						
18	.	43					23	20						
19	.	46					20	26						
20	.	120							20	20	20	20	20	20
21	.	70							16	16	16	16	16	16
22	.	17												17
23	ETC.	15												15
TOTAL METRES			98	107	106	102	103	92	66	66	58	57	55	78

The Planner sits at a screen and manipulates a bar chart made up of strings of numbers (Figure 2).

The numbers are metres (or square metres or shifts) which are to be mined in each month in each working place; e.g., a tunnel or stoping panel.

The Planner first selects the set of workplaces to be mined by one team and schedules the work, bearing in mind the required sequence. He knows the total monthly metres which can be broken by the team working in that particular ground, with the specified number of people, and he manipulates the schedule so as to get the correct monthly totals in the bottom row of the screen.

He can do two things to the schedule:

- Change individual numbers by direct overtyping of the displayed numbers (making use of the full-screen formatting facility). Totals are re-calculated when he presses the ENTER button.
- Move strings of numbers to the left or right using commands specifying, for example, that activity 2 must start after so many units along activity 1. Any activities already linked to follow 2 will also move, and totals at the bottom of the screen will be re-calculated.

The screen can be “moved” to view all parts of a large schedule.

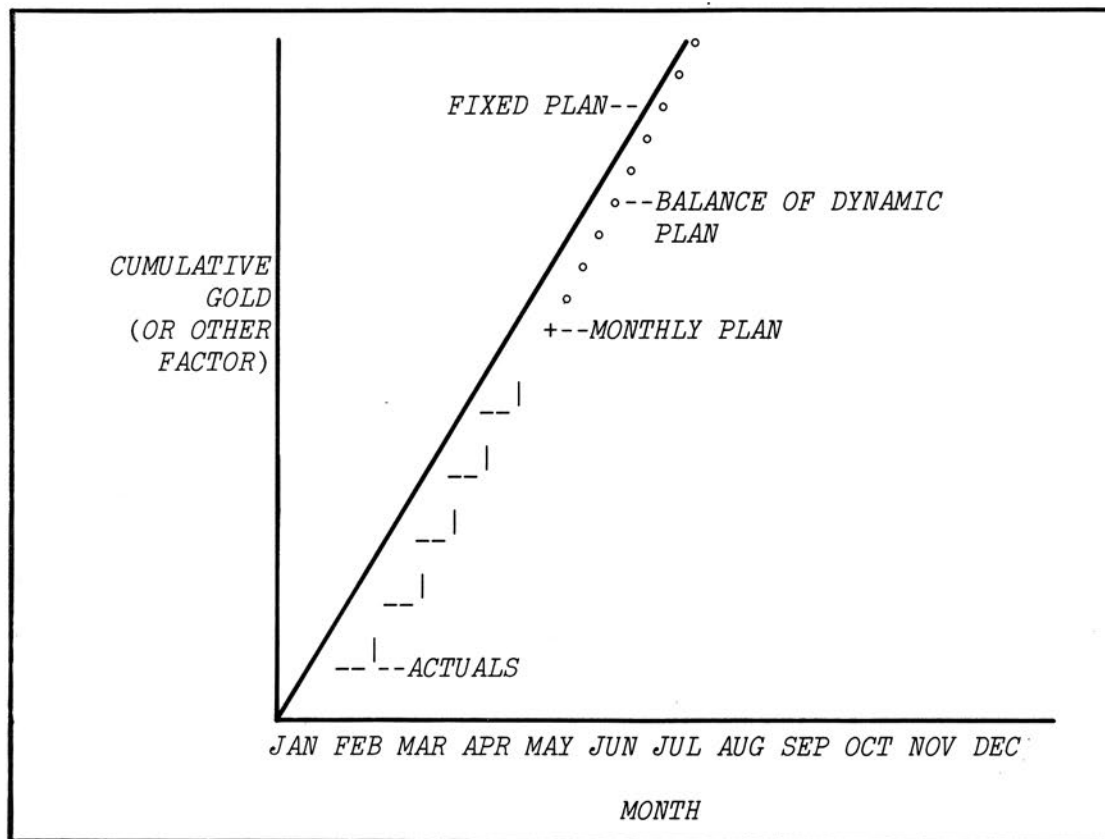
A simple auto-scheduling algorithm has recently been added which manipulates the schedule to achieve the required monthly totals given priorities for the working places.

Having scheduled the work for one team for, say, 2 years, he then selects the next team.

Finally, he can call for various reports which print out the schedule, together with an evaluation (tons, gold, gold grade, etc.), totalled at various management levels.

When the plan has been approved, the Planner saves two copies; one is the fixed target which will remain unchanged for twelve months and the other is the dynamic plan which will be updated with monthly plans and actuals, and will be re-scheduled to keep on target. Variance reports of planned against actual work can then be produced as line printer graphs and tables (Figure 3).

Figure 3
A Variance Report



Other sub-systems which have been added to the PROMIS system are:

- Area contribution, which calculates costs and returns for any selected sub-set of workplaces.
- Manpower planning, which calculates the labour required to execute the plan. The labour plan is then passed to a large IMS Personnel system (HURIS).
- Loco efficiency system, which estimates the number of locomotives required to transport the rock underground.

In future it is intended to calculate stores and equipment requirements from the production plan and then to produce draft budgets.

How the system is being used at present

The system is being used on six of the nine gold mines, a total of 25 shafts. The way it is used varies from mine to mine. The extremes are:

- Used during the planning period only, to develop and evaluate the two year plan, then left till the following year.

- Used to develop the plan which is then updated each month with actuals and monthly plans. Monthly cumulative variance reports are produced. The dynamic version of the plan is regularly re-scheduled to get back onto the target as represented by the fixed version.

Factors affecting the rate of progress towards the second situation are:

- Management commitment to medium-term planning and control
- Presence on the mine of a keen co-ordinator
- Number of people on the mine who know the system
- Training and support by INFOGOLD
- Ease of use of the system

We are now entering a consolidation phase and will be concentrating on those activities and improvements which assist Mine Management in gaining the maximum benefit from the system.

Development story—How to learn by getting it wrong

Some four and a half years ago it was decided that some computer assistance should be given in the planning and control process. The new, to us, and highly experimental language APL was chosen (VSAPL under TSO), to at least prototype something.

Setting out with an abundant confidence only matched by lack of experience, an interactive scheduling screen was designed, supported by a simple (and in the early stages unworkable) algorithm [Ref. 1].

We began with a workspace size of two megabytes, with all functions and data in the workspace, and were mildly amused by the way D.P. people kept fussing about files. All went well until the program began to catch on and the number of simultaneous users rose to four or five. We were causing havoc on the machines (two IBM 158 M.P.'s).

Panic set in and two people spent three months modifying the system to run in one megabyte and to make it more efficient by re-writing some of the more heavily used code. One young trainee changed some key functions, written by his superior, to run twenty times faster (he came within an ace of dismissal).

These changes relieved the situation and for a time all was calm. Additional sub-systems were then required, e.g., variances, mine totals and area contribution. These could not be fitted into one megabyte; system design using workspaces alone became a nightmare. We then invested some six man months in finding out how to use files and settled on *VAR* files, in which an APL variable can be written onto a file record. The files were either fixed length direct access or variable length sequential access.

This was a breakthrough (another wheel invented) and we began to see the possibilities for large system design using a combination of workspaces and files.

Our scheduling sub-system still had all functions and data in the workspace and a number of users were getting *WORKSPACE FULL* as they tackled larger schedules.

The number of simultaneous APL users was creeping up and was reaching 13 at peak times. We were again punishing the machine, which by now was an IBM 3032. (It should be noted that the machine was also running a large amount of non-APL work).

At this stage we came across I.P. Sharp software and began a long fight to justify it as a replacement for VSAPL.

A trial installation was agreed to and this spurred us to carry out, on the existing software, the structural changes necessary to allow the loading of sub-sets of functions and data into the workspace. This was a task we had been putting off due to inefficiencies in the file system. About five man months were invested in this conversion and the system was released for use by customers. It required 1/3 to 1/2 megabyte of available workspace. Response was slower than with the previous version and some reports ran for thirty minutes (compared with two to four minutes) due to the amount of Input/Output. The system was accepted by the customers because we had added full screen formatting for the scheduling and data change screens, and because there was the promise of better response under the new software in the future.

This design raised the limit on the number of workplaces in the schedule (since only a sub-set is read in at any one time) and the addition of further sub-systems has become an easy matter. This system, together with all others, was converted by I.P. Sharp to run on their software, which has now been installed and accepted. Average response on the PROMIS Planning System is two to three times faster on SHARP APL and report calculation times dropped to two to three minutes. However, these reports are now being run in concurrent batch, so that the user sees a three second response, the time it takes to start the report running.

Machine resource usage is also lower, particularly Input/Output which has been reduced by some 85%.

A systems architecture for future development

The development of PROMIS and its allied systems has taken us some way towards the architecture illustrated in Figure 4.

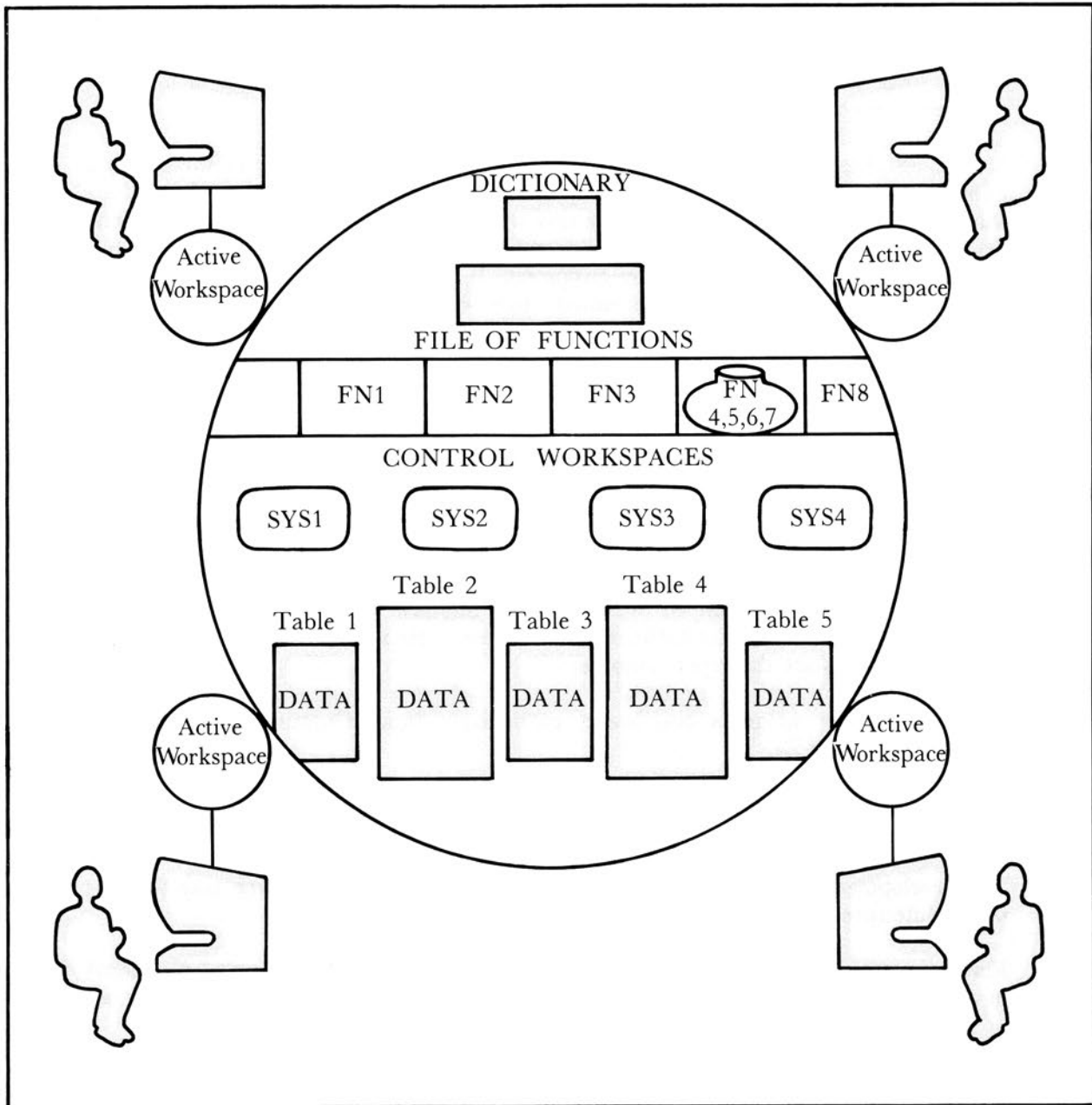
Functions are stored on one or more files with one function (or a group of functions) in a package per file component. Most data is stored on files in a normalised format.

As far as possible the data tables are maintained (add, change, delete, search, sort, display) by a set of general functions which use the data dictionary to get the required information about the particular table being modified.

A set of control workspaces is stored, one for each application, and these contain one or a small group of functions needed to get started (Sys 1, Sys 2, etc.).

Figure 4

Intended Framework for APL Systems At INFOGOLD



Future use of APL

We foresee an increasing role for APL because of its well known advantages:

- Fast development
- Ease of maintenance
- Ease of change
- Shorter staff training

Areas of expansion in the near future are expected to be:

- Geology
- Strategic Planning
- Metallurgical Planning and Control
- Management reporting using data stripped from IMS data bases

Reference

1. Corrie, W.J. *A Scheduling Algorithm*. Paper presented to Anglo American Corporation Group Computer Conference, 1981.

INTERACTIVE DECISION MAKING: CREATING DECISION TREES WITH APL*

**E.A. Boyd
Lincoln Laboratory
Massachusetts Institute of Technology**

Decision making has developed a long way from the age of intuitive decisions. The advent of business oriented mathematical techniques, high speed computing machinery and its ability to organize and process data, and the continuing need to cut costs in an increasingly competitive world have brought decision making to the state of a science. But this science becomes an art only when the tools are used effectively to reach proper decisions, and the key to effective use is a matter of organization.

The best organizational tool available to the decision maker is the decision tree. On the surface, it is deceptively simple. An individual confronted with a decision tree for the first time should have little difficulty understanding it. Yet, it is the very simplicity of the decision tree which makes it such a powerful tool. By reducing complex decision problems to a series of steps and presenting the decision process in an understandable format, the decision tree exposes a problem for analysis. In this way, carefully considered rational decisions can be made. Decision trees have the additional advantage of presenting an entire decision problem at a glance, making them perfect for summarizing and reporting results to others.

However, decision trees have the drawback of being bulky and difficult to handle. Much of the time spent working with them involves drawing and redrawing them. With this task allocated to the computer a world of flexibility becomes available to the decision maker.

This paper describes a workspace designed to manipulate decision trees effectively. The work was inspired by APL's array handling facilities, and the completed workspace owes its very existence to APL. The design of the workspace is presented along with the algorithm for actually constructing the tree, and the role of APL in the design is discussed.

* This work was sponsored by the Department of the Air Force.

The U.S. Government assumes no responsibility for this presentation.

The decision tree

Two examples of completed decision trees are pictured in Figures 1 and 2. The quads represent decision nodes and the circles chance nodes. One or more paths extend from each node. These paths represent alternative events which may occur at each node. At the chance nodes the path taken is a random event, while at decision nodes the path taken is left to the decision maker. Paths extending from chance nodes bear probabilities representing the assessed likelihood of a particular event occurring.

The column labelled *TERMINAL VALUES* contains the amount which will be gained or lost at the completion of a given path. The terminal values will often be dollar amounts, though they need not be. The column labelled *UTILITIES* is optional and contains the assessed relative value of each of the possible outcomes.

The function of the decision tree is straightforward: determine the optimal path to follow from each decision node. Since chance nodes exist, no single optimal path may be defined. Further, the optimal path to choose at each decision node will depend upon the expected value of each possible path. In the completed tree, paths which extend from decision nodes and are not optimal are blocked with two vertical lines. The completed tree thus outlines the entire series of steps composing a larger decision process and exposes the optimal path extending from each decision node.

The format

In designing a workspace to manipulate decision trees the actual format of the decision tree was a major concern. A number of criteria had to be met in order to construct a useful workspace. The formatted tree had to be easy to understand at a glance. Information contained within the tree had to be easily accessible to the user so that initial data entry could be kept to a minimum and changes to the tree could be made with ease. Finally, a design which would handle trees of all shapes and sizes had to be developed.

All of these criteria were satisfied by the adoption of two conventions for construction of decision trees by the user. Technically, these conventions can be stated as follows:

- 1) For all paths taken through a decision tree, the same number of decision/chance events are encountered.
- 2) For all paths taken through a decision tree, the n-th event is of the same type (a decision or chance event).

In practice, these conventions can be adhered to with no conscious recognition on the part of the user. The result is a decision tree with distinct visual columns—each column representing the presence of a new step in the decision process.

These columns can in turn be used to reference the location of data within the tree. Initial data entry is made by entering each set of data column by column, top to bottom. A specific piece of data may be changed by referencing the type of data, its column, and how far down the column it resides. Each individual event in the decision process need not be defined as a decision or chance event since defining each column will serve an equivalent function.

Other formatting conventions exist, but are invisible to the user. Paths extending from

decision and chance nodes extend from a vertical line passing through the node rather than extending directly from the node itself. This convention serves numerous purposes. It visually emphasizes the columns used for referencing data. It also eliminates the need to draw lines that are neither horizontal nor vertical. This would become necessary if each path extended directly from a decision or chance node.

An additional advantage to this approach is that it leaves space immediately to the right of each node in the tree. This space can in turn be used to hold descriptive information associated with each node. The only time this space is unavailable is when a single path extends from a node, and at these times the descriptive information is unnecessary.

Each column is adjusted to be wide enough to hold the longest string of characters in that column. This leads to columns of varying width, but minimizes the horizontal space required by the tree. The user is offered the option of setting a minimum column width, and by choosing a large enough value the columns may have a standardized width. The sacrifice is that additional memory is required.

Data input

Another extremely important part of the workspace is data input. To be effective, it was felt that data entry had to be kept as simple as possible. This was not a trivial task given the extremely general nature of decision trees. Certain types of data entry could not be reduced. Comments, probabilities, terminal values, and utilities must be entered by the user. Thus, the objective was to minimize data relating to the structure of the tree, and to simplify the way in which all data are entered.

A major simplification came from the formatting conventions discussed in the previous section. The visual columns and horizontal lines which appear in the decision tree offer a very natural division of the tree. For initial data entry, terminal values and utilities are entered as a vector with the order of input the same as that in which the values appear from top to bottom in the completed tree. Comments and probabilities are entered in the same way—one column at a time.

After considering a number of ways for describing the structure of the tree, a method requiring two additional inputs was decided upon. The first is a character vector filled with D's and C's representing whether each column is a decision or chance column. Again, the formatting conventions helped simplify the process. The second contains the number of paths extending from each node in the tree and is entered in a fashion analogous to the comments and probabilities.

A set of functions was written to support data entry. The most important of these, *INPUT*, requests each piece of necessary information, performs checks for consistency on the entered data, and contains a series of default conditions for expediting data entry. A series of separate functions was also developed for this purpose, but designed primarily for use as function calls in user developed programs. For work in an interactive environment, *INPUT* serves adequately for all purposes.

Functions were also written for updating data in a previously defined tree. These functions greatly facilitate the speed with which sensitivity analyses can be performed, and in doing so create an interactive decision making environment.

Constructing the tree

Constructing a completed decision tree is different from initially conceiving and sketching the same tree. A decision process flows from an initial point with branches extending outward. This means that a tree will be physically sparse at the beginning of the decision process but dense near the end. A common difficulty when initially sketching trees is failure to leave adequate room for the remaining branches. This problem can be easily overcome when formatting a completed tree. Building the tree backwards (right to left on a physical page) allows the dense portion of the tree to be constructed first, assuring ample space for all of the tree. This approach is a major driving factor in the construction of the completed tree.

The flow of the construction is pictured in Figure 4. The first step is to determine the optimal path to follow from each decision node. This involves computing and interpreting the values found in the decision tree to the immediate right of each of the decision and chance nodes. When this is completed all of the information necessary for constructing the finished tree is available. Once numeric data such as the probabilities, terminal values, and utilities have been converted to character strings the problem is reduced to putting desired characters in their proper position in the matrix which will hold the decision tree.

The construction process begins by reshaping the terminal values and utilities into the form they will take in the finished tree. This task is accomplished with a straightforward application of the reshape, expansion, and indexing operators. The vertical space between each value is constant and may be specified by the user through a global variable *SPACE*.

Each visual column (not column of characters) is then created one at a time working from right to left as the columns appear in the completed tree. The algorithm lends itself to a recursive approach because the location of the decision and chance nodes of one column affects the structure of the adjacent column. The important information required for constructing a visual column is the index of the row in the completed matrix in which each character string will reside. A variable *OLDNODES* is used to contain the indices of the rows of the nodes on the right side of the column being constructed (the left side of column just completed). Using user supplied information on the structure of the tree, the indices of the rows of the nodes on the left of the column being constructed are computed and stored in a variable *NEWNODES*. Together, the values within *OLDNODES* and *NEWNODES* are used to locate the relative position of every character string in the visual column. The width of the visual column is calculated by determining the length of the longest character string it must contain. Again using the reshape, expansion, and indexing operators the visual column can be easily completed and laminated to the previously completed portion of the tree. Storing *NEWNODES* in *OLDNODES*, the process is repeated until the tree is complete.

Conclusions

The algorithm presented here appears very basic because it is. The code for actually constructing the tree is compact, consisting of about 150 lines. But the simplicity of the design and execution can largely be attributed to APL. This algorithm could be programmed in more conventional languages, but it does not lend itself to such an approach. The code required in other languages to perform the same algorithm could

easily become so cumbersome that the project would not be worthwhile. The functions of reshaping, laminating, and expanding matrices cannot easily be mimicked in other languages.

But even more, the algorithm itself owes its initial conception to APL. Developing an algorithm is a process of consciously bringing available tools together to solve a specific problem. Most languages have so much in common that an initial algorithm may be defined without knowledge of the language in which the solution will be programmed. APL is unique in that it offers a fundamentally different conceptual approach to programming. In most instances it leads to more compact code. But the true value of APL becomes apparent when it inspires a simple solution to an otherwise difficult problem.

Figure 1
Time Minimization Problem

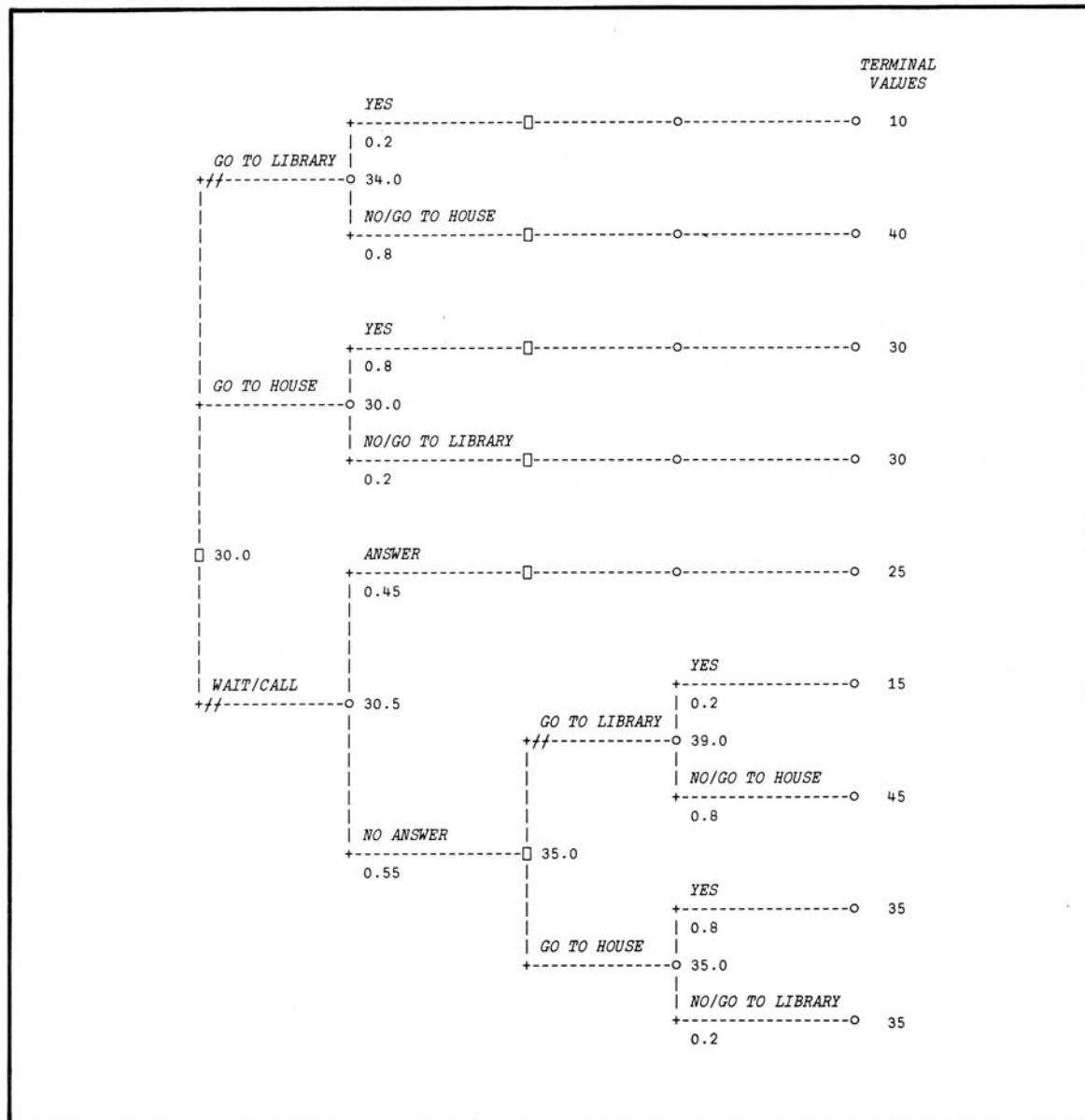


Figure 2
Machine Repair Problem

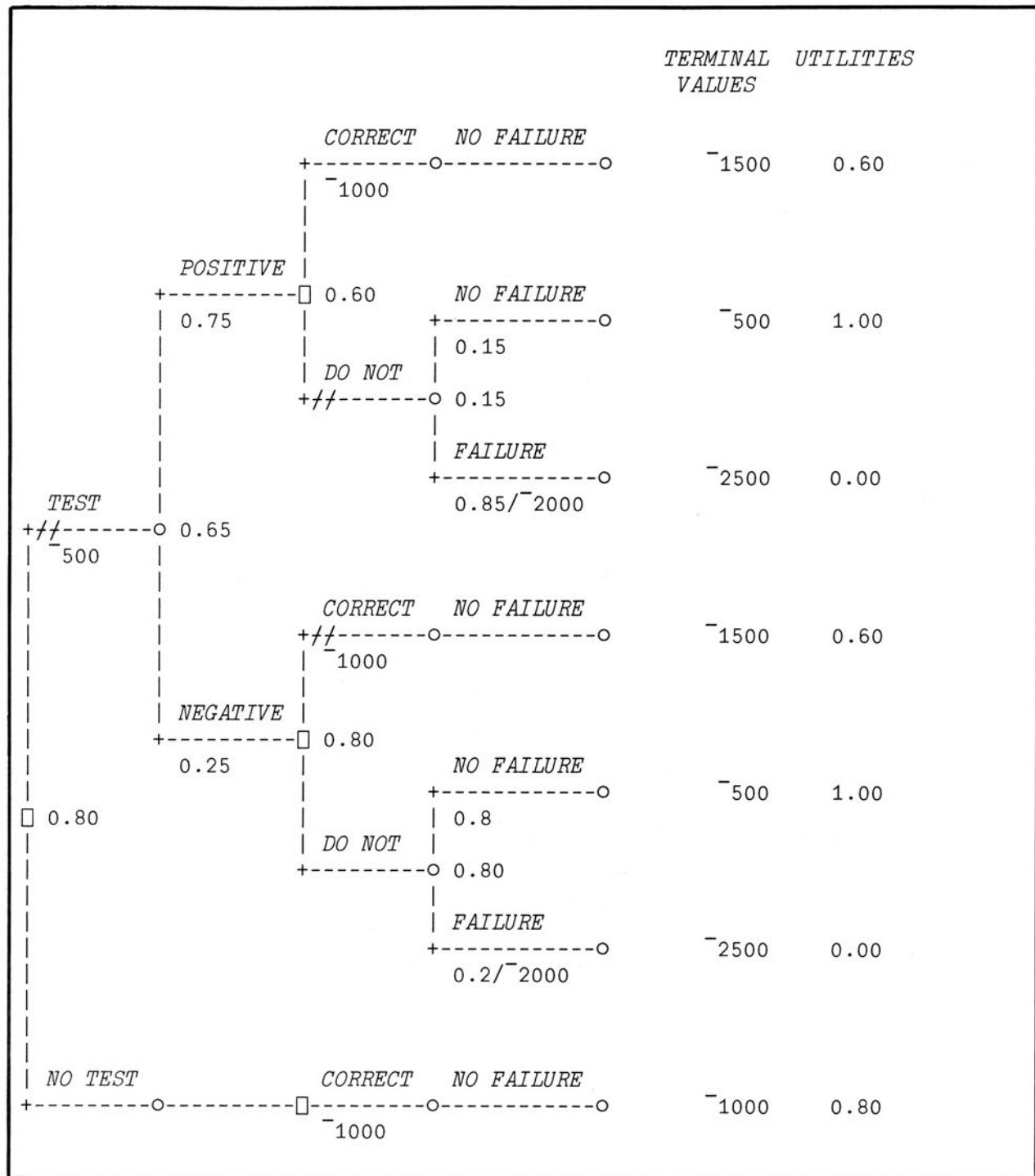


Figure 3
Data Input for Figure 2

INPUT

ENTER DCV (DECISION/CHANCE CHARACTER VECTOR):

□: 'DCDC'

ENTER TERMINAL VALUES (TOP TO BOTTOM):

□: -1500 -500 -2500 -1500 -500 -2500 -1000

ENTER UTILITIES (TOP TO BOTTOM; NO UTILITIES ENTER 10):

□: .6 1 0 .6 1 0 .8

ENTER LM (LINE MATRIX; TOP TO BOTTOM, LEFT TO RIGHT):

□: 2 2 1 2 2 1 1 2 1 2 1

ENTER PM (PROBABILITY MATRIX; TOP TO BOTTOM, LEFT TO RIGHT):

□: 1 1 .75 .25 1 1 1 1 1 1 1 .15 .85 1 .8 .2 1

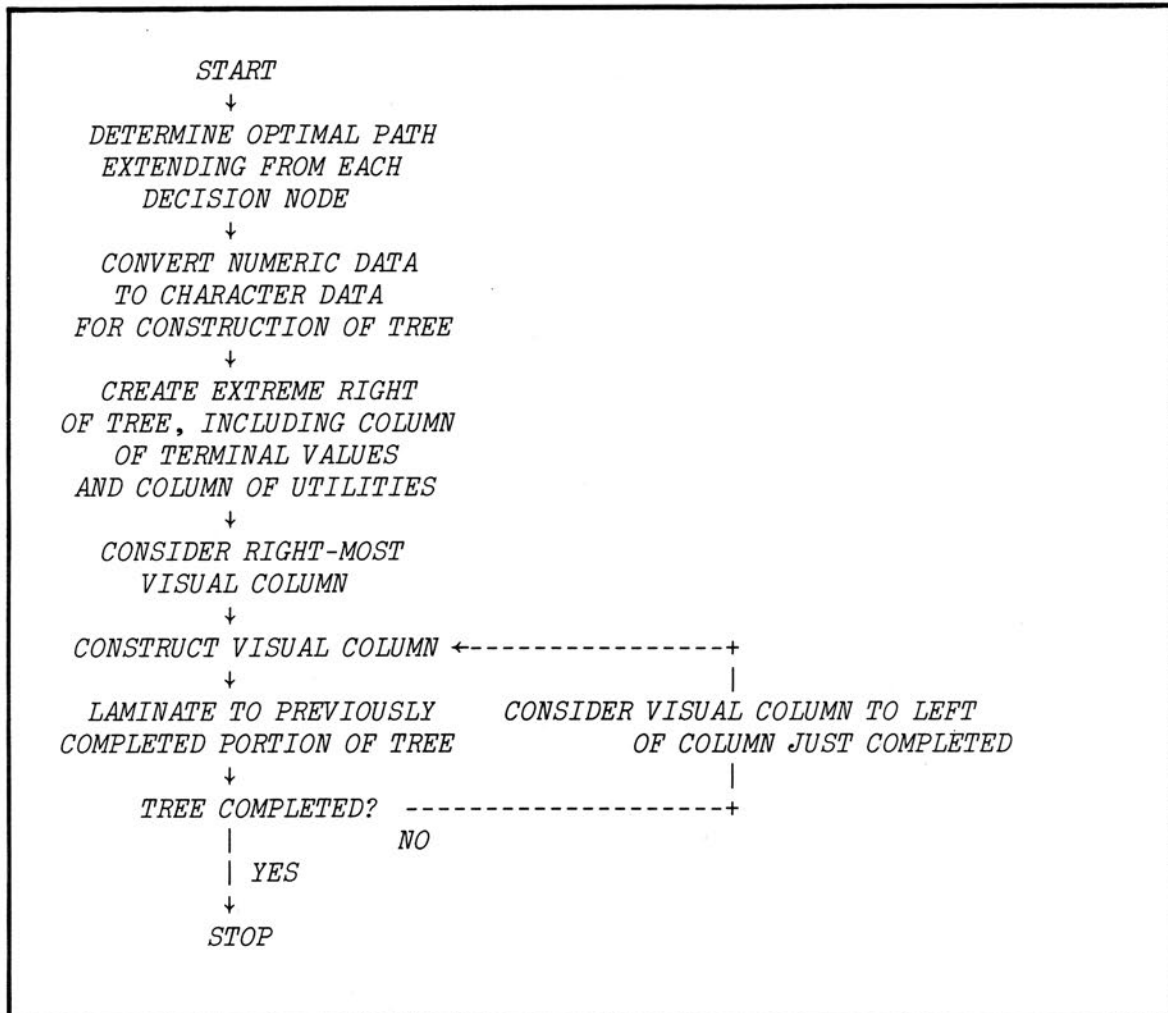
ENTER CM (CHARACTER MATRIX; TOP TO BOTTOM, LEFT TO RIGHT):

□: '/TEST/NO TEST/POSITIVE/NEGATIVE//CORRECT/DO NOT/CORRECT
/DO NOT/CORRECT/NO FAILURE/NO FAILURE/FAILURE/NO FAILURE/NO FAI
LURE/FAILURE/NO FAILURE'

ENTER NM (N MATRIX; TOP TO BOTTOM, LEFT TO RIGHT):

□: '/-500////-1000//1000//1000//2000//2000/'

Figure 4
Program Flow



APL AS A FUNCTIONAL PROGRAMMING LANGUAGE

Eugene McDonnell
I.P. Sharp Associates, Inc.
Palo Alto, California

Abstract

This paper gives an elementary introduction to the notion of functional programming, and gives examples of how to define APL functions in a functional programming style.

Functional programming

The term *functional programming* has come into use in recent years to describe a style of programming in which the application of functions to arguments is the only permitted form. The interest in functional programming comes from several sources, but one of the chief ones is a dissatisfaction with conventional programming languages, such as ALGOL, BASIC, COBOL, FORTRAN, and PL/1. In a key paper [Ref. 1], John Backus, the father of FORTRAN, gave wide exposure to these ideas. These languages are thought to have basic defects in that they essentially process data one element at a time, and use assignment as a fundamental control form. Backus feels that APL is exempted from the first defect, because of its array orientation, but that it is still imperfect because of its use of assignment. By adhering to certain restrictions, however, it is possible to make a purely functional language out of APL. These restrictions are:

- don't use assignment
- use the direct definition form of function definition
- don't use global variables

An example

The problem is to make a table which shows all possible ways of choosing M objects from N different objects. For example, if we label the objects a, b, c, d, and e, and want to tabulate all the ways of choosing three of these, we would be satisfied with the result:

```
a b c
a b d
a b e
a c d
a c e
```

```

a d e
b c d
b c e
b d e
c d e

```

It is often more convenient simply to make a table of the indices which can be used to select the objects, as follows:

```

1 2 3
1 2 4
1 2 5
1 3 4
1 3 5
1 4 5
2 3 4
2 3 5
2 4 5
3 4 5

```

The idea underlying the functions we will define is to create an incidence matrix (a matrix of ones and zeros) whose rows show the positions of the chosen objects, and to produce the matrix of indices from this incidence matrix. Thus, for our 3-out-of-5 problem we would want the incidence matrix:

```

1 1 1 0 0
1 1 0 1 0
1 1 0 0 1
1 0 1 1 0
1 0 1 0 1
1 0 0 1 1
0 1 1 1 0
0 1 1 0 1
0 1 0 1 1
0 0 1 1 1

```

Putting an APL solution together

A programmer might observe that the desired incidence matrix could be extracted from the rows of a complete truth table, as generated by an expression such as $\mathbb{Q}(\omega\rho 2)\top_1 2*\omega$. For example, the truth table of order 2 looks as follows:

```

0 0 0
0 0 1
0 1 0
0 1 1
1 0 0
1 0 1
1 1 0
1 1 1

```

The incidence matrix for the 2-out-of-3 case could be obtained from this by selecting those rows which summed to 2, namely the fourth, sixth, and seventh rows:

```
0 1 1
1 0 1
1 1 0
```

At this point it might be noticed that using this incidence matrix will not yield the indices in lexical order. Instead of:

```
1 2
1 3
2 3
```

we would have instead:

```
2 3
1 3
1 2
```

To remedy this, we could do several things. Perhaps the simplest is to complement the truth table before doing the selection, giving us:

```
1 1 1
1 1 0
1 0 1
1 0 0
0 1 1
0 1 0
0 0 1
0 0 0
```

Now when we select the rows summing to 2, they come out in the desired order:

```
1 1 0
1 0 1
0 1 1
```

and the corresponding index rows are as we want them, too:

```
1 2
1 3
2 3
```

To put this thought together in an APL expression is for the experienced APL programmer but the matter of a moment: the truth table is given by $\sim\Phi(\omega\rho 2)\top 1 2\star\omega$. (Actually, this truth table, being in 1-origin, has the first row in the last place, but that doesn't affect our use.) Now, to select the rows which sum to the desired amount, we must perform a sum reduction of the rows of this matrix, and a comparison with the desired value, in order to produce the selector mask. Here we run into a quandary. We would prefer not to have to compute the truth table twice, for aesthetic reasons, as well as from considerations of efficiency. If we were allowed to use assignment we could create a temporary variable T and write something like:

```
( $\alpha=+/T$ ) $\nearrow T\leftarrow\sim\Phi(\omega\rho 2)\top 1 2\star\omega$ .
```

Lacking assignment, what can we do? A little thought will reveal that we are dealing with a function of α and our modified truth table. This suggests that we define a function, say K , which takes α as one argument and the modified truth table as the other. If we then write:

$$\alpha K \sim \Phi(\omega \rho 2) \tau_{12} * \omega$$

it shouldn't take too long to define:

$$K \Diamond (\alpha = +/\omega) \neq \omega$$

Having done this, we can try the function out immediately:

$$\begin{array}{c} 3 \ K \sim \Phi(5 \rho 2) \tau_{12} * 5 \\ 1 \ 1 \ 1 \ 0 \ 0 \\ 1 \ 1 \ 0 \ 1 \ 0 \\ 1 \ 1 \ 0 \ 0 \ 1 \\ 1 \ 0 \ 1 \ 1 \ 0 \\ 1 \ 0 \ 1 \ 0 \ 1 \\ 1 \ 0 \ 0 \ 1 \ 1 \\ 0 \ 1 \ 1 \ 1 \ 0 \\ 0 \ 1 \ 1 \ 0 \ 1 \\ 0 \ 1 \ 0 \ 1 \ 1 \\ 0 \ 0 \ 1 \ 1 \ 1 \end{array}$$

So far, so good. The next task will be to convert the rows of this incidence matrix, to give the column indices corresponding to the 1's in each row. There are many different ways to do this, but we'll choose a method that helps reinforce the point we made before on how to do without assignment.

The essential idea is to generate as many copies of the set of indices as we need, and then select the desired elements using the ravel of the incidence matrix. One way to generate the desired indices is to generate one set of indices (by $\iota \omega$), reshape this to the shape of the incidence matrix, and ravel this. Thus, in the 2 out of 3 case, we could write:

$$\begin{array}{c} , 3 \ 3 \rho 1 \ 3 \\ 1 \ 2 \ 3 \ 1 \ 2 \ 3 \ 1 \ 2 \ 3 \end{array}$$

We can then use the ravelled incidence matrix to select the desired index elements from this. We would get:

$$\begin{array}{c} 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1 / 1 \ 2 \ 3 \ 1 \ 2 \ 3 \ 1 \ 2 \ 3 \\ 1 \ 2 \ 1 \ 3 \ 2 \ 3 \end{array}$$

Since the number of elements selected from each row is the same, this last result can be reshaped easily into the final result:

$$\begin{array}{c} 3 \ 2 \rho 1 \ 2 \ 1 \ 3 \ 2 \ 3 \\ 1 \ 2 \\ 1 \ 3 \\ 2 \ 3 \end{array}$$

However, we have the same problem again that we had before, because we want to use the incidence matrix as an argument to compression, and we also want to use the shape of the incidence matrix as an argument to reshape.

The function we would have if we used assignment would look like:

$$U \diamond ((\alpha! \omega), \alpha) \rho (, S) / , (\rho S \leftarrow (\alpha = + / T) / T \leftarrow \diamond (\omega \rho 2) \tau 1 2 * \omega) \rho 1 \omega$$

In this, the incidence matrix is given the name S , and then we use its shape and its ravel. Since we don't want to use assignment, we must create another function, say I , which uses the incidence matrix as one argument, and the set of indices as another. The function I looks like this:

$$I \diamond (, \alpha) / , (\rho \alpha) \rho \omega$$

We could then write:

$$(\alpha K \sim \diamond (\omega \rho 2) \tau 1 2 * \omega) I 1 \omega$$

which, when used with the arguments 3 and 5, would yield:

1 2 3 1 2 4 1 2 5 1 3 4 1 3 5 1 4 5 2 3 4 2 3 5 2 4 5 3 4 5

It is a simple matter to reshape this: the number of rows is $\alpha! \omega$ and the number of columns is α . The entire $COMB$ function now would look like this:

$$COMB \diamond ((\alpha! \omega), \alpha) \rho (\alpha K \sim \diamond (\omega \rho 2) \tau 1 2 * \omega) I 1 \omega$$

and

3 COMB 5

1 2 3
1 2 4
1 2 5
1 3 4
1 3 5
1 4 5
2 3 4
2 3 5
2 4 5
3 4 5

The set of functions we have now produces its result without using assignment, and without variables. In practice, we would find that the generation of the entire truth table produces an intermediate value which uses a fair amount of workspace, first because of the rows which are generated only to be discarded, and second because the $\alpha \tau \omega$ function produces an integer result when used with integer arguments, and so, even though all the elements of the result are 1's and 0's, these are stored as integers, not Boolean values. Some experimentation may convince you that the function J below produces an incidence matrix of the desired kind which has only the required rows, and which is stored as a Boolean matrix:

$$J \diamond (1, (\alpha - 1) J \omega - 1), [1] 0, \alpha J \omega - 1 \diamond \alpha \in 0, \omega \diamond (1, \omega) \rho \alpha = \omega$$

This doubly-recursive function does its work by dividing the problem into two parts, and concatenating the first part result on top of the second part result. The first part is obtained by concatenating a 1 in front of the matrix which has $\alpha-1$ 1's out of a total of $\omega-1$, and the second part is obtained by concatenating a 0 in front of the matrix which has α 1's, also out of a total of $\omega-1$. The terminating condition for the function is given if α is 0 or if it is equal to ω , and the result in that case is a 1 by ω matrix which is either all 0's (if α and ω are unequal), or all 1's (if α and ω are equal).

A further improvement could be made to the *COMB* function if we notice that the desired set of indices could be obtained without reference to the incidence matrix by an expression such as $(\omega \times \alpha! \omega) \rho 1 \omega$. For example,

$$\begin{array}{c} (3 \times 2! 3) \rho 1 3 \\ 1 \ 2 \ 3 \ 1 \ 2 \ 3 \ 1 \ 2 \ 3 \end{array}$$

This would allow us to eliminate the function *I*. Putting these two thoughts together gives us the functions *COMB* and *J* as follows:

$$\begin{array}{l} COMB \diamond ((\alpha! \omega), \alpha) \rho (, \alpha J \omega) / (\omega \times \alpha! \omega) \rho 1 \omega \\ J \diamond (1, (\alpha-1) J \omega-1), [1] 0, \alpha J \omega-1 \diamond \alpha \in 0, \omega \diamond (1, \omega) \rho \alpha = \omega \end{array}$$

A more elaborate use of APL in a functional programming style may be found in reference [2].

Functional programming enhancements to APL

Several of the additions proposed for APL would increase its suitability as a functional programming language. Among these are many new operators and an indexing function.

Backus writes that a proper set of operators is required in a functional programming language. He claims that APL, which has only the four operators of reduction, inner product, outer product, and axis, doesn't have enough of these forms. However, the principal thrust of the work of K.E. Iverson over the last several years has been in the direction of increasing the number of operators in the language. Already, in SHARP APL the three composition operators \circ $\circ\circ$ $\circ\circ\circ$ have been introduced, in a limited fashion, and several more have been discussed and will undoubtedly be implemented one day. For details of these operators, see references [3 4 5 6].

Indexed assignment is frequently used to make modifications to a part of an array. If assignment is not permitted in functional programming, some alternative method of achieving this would be necessary. One proposal [Ref. 3] to do this uses a new Indexing-function, together with one of the composition operators. Thus, to replace with New-values the elements in an Array selected by an Index, one would write:

New-values Index \circ Indexing-function Array

References

1. Backus, John. "Can Programming be Liberated from the von Neumann Style? A Functional Style and its Algebra of Programs." *Communications of the ACM* 21, 8 August 1978, pp. 613-641.
2. McDonnell, E.E. *The Four Cube Problem, A Case Study in BASIC, APL, and Functional Programming*. Palo Alto: APL Press, 1981.
3. Bernecky, Bob, and K.E. Iverson. "Operators and Enclosed Arrays." *Proceedings of the 1980 APL Users Meeting*. Toronto: I.P. Sharp Associates, pp. 319-331.
4. Iverson, K.E., and Arthur T. Whitney. "Practical Uses of a Model of APL." *APL82 Conference Proceedings*. W.H. Janko and W. Stucky, eds, in *APL Quote Quad* 13, 1 September 1982, pp. 140-145.
5. Iverson, K.E. "Operators." *ACM Transactions on Programming Languages and Systems* 1, 2 October 1979, pp. 161-176.
6. Iverson, K.E. "Operators and Functions." *RC7091*, IBM Corp., 1978.

A MODELLING TOOL FOR EVALUATING PROPOSED OIL & GAS EXPLORATION EXPENDITURES

**Terry B. Peterson
Senior Staff Financial Analyst
Corporate Planning and Business Development
Canadian Occidental Petroleum Limited
Calgary, Alberta**

Introduction

Canadian Occidental Petroleum Ltd. is a medium sized company involved in chemicals, mining, and oil and gas production. Currently, the Oil and Gas Division contributes about 60% of the company's total revenue.

CanadianOxy's financial planning group is small, but we play an active role in providing top management with analysis of the financial effects of their proposed actions. To accomplish this, we have developed several computer-based modelling tools, all of them written in APL. Among these is our financial planning model which, since first written in 1977, has been continually refined and updated. The model enables us to prepare *pro forma* financial statements for each division of the company, using built-in functions to compute royalties and income taxes for six different jurisdictions across Canada, U.S. and South America. These files are then consolidated by the model, recalculating income taxes where applicable, to arrive at the total company forecast.

A simpler application was the development of a system for monitoring each division's capital expenditure budget using I.P. Sharp's MABRA. This system enables us to quickly sum capital expenditures by division, level of expenditure or other attribute, and to list projects by priority.

Two other applications have been developed that relate primarily to the Oil and Gas Division. Both of these applications are used to evaluate different sets of assumptions so we use I.P. Sharp's STARS system to facilitate the handling of the various cases. The first of these is a model of the pricing mechanism agreed upon by the Canadian Federal and provincial governments for pricing of domestic oil and natural gas. Based on a given world oil price forecast, and some assumptions regarding Canada's response to that forecast, we can generate a schedule of the expected Canadian prices.

The second application we developed using the STARS system is a model for evaluating proposed oil and gas exploration expenditures. This is the model I will deal with in detail below.

The application

Once a year each of the company's divisions is required to submit a ten year plan which is then consolidated into a strategic plan for the company. Each divisional plan must include forecasts of expected revenues, operating costs, and capital expenditures. Also, the plan must be supported by an economic analysis indicating that the incremental investment required in the plan will earn the company's target rate of return—given certain stated assumptions. If management thinks the assumptions are realistic, the plan is approved.

In CanadianOxy's Chemical Division, the task of putting together the plan may not be easy, but it is relatively straightforward compared to that faced by the Oil and Gas Division. Forecasts of the future prices and demand volumes for individual chemical products are available from several sources. On the basis of the division's forecast of its market share and its known plant capacity, one can reasonably determine the point in time that capital expenditures will be needed and whether they can be justified by the associated incremental revenues.

Forecasting and justifying future capital expenditures in the oil and gas exploration business is much more difficult. One can not assume that a given capital expenditure will increase production capacity, as a plant expansion in the chemical business would. There is a very good chance that the exploration expenditure will result in a dry hole and no additional revenues will be available to generate the required rate of return. Even if the exploration is successful and the drilling results in a producing well, additional expenditures will be required to properly develop the discovered reserves and transport the product to market. The level of these additional expenditures will depend on the amount of reserves discovered, whether the product was oil or gas, the location of the discovery relative to existing transmission facilities, etc. The timing of the expenditures will also depend on the product discovered. For example, there is presently an oversupply of available gas production so no additional development expenditures would be scheduled for a gas discovery until just before the gas was expected to marketed.

Other considerations important to the economics of the plan are the forecast production volumes and the amount of the total exploration budget spent on each expenditure category. The initial production available is related to the amount of reserves discovered, but the production can be expected to decline exponentially as the reserves are depleted. In Canada, the different categories of expenditure receive very different tax treatment, so it is also necessary to determine the allocation of the total dollars spent.

The model

For those who are not familiar with I.P. Sharp's STARS facility, each system created has an integral data file which contains the default values for all the input variables used. Figure 1 indicates the variables used in the exploration model and the default values assigned to them.

Probably the most important variables in the model are the finding costs. Finding costs are determined by dividing the total dollars spent on exploration during a particular year, either by one company or the industry, by the total reserves discovered as a result of that expenditure. The calculation takes into account the probability of drilling a dry hole because all exploration expenditures, including those that result in dry holes, are

included. The annual escalation of the finding cost reflects both the inflationary component and the increased difficulty in finding reserves as an area is continually explored.

Another important variable is the probability of finding either oil or gas when exploration is undertaken. Again, this number can be determined from the company's own past experience or from a study of the industry findings. However, this probability will presumably change if the company directs its expenditures exclusively towards finding one of the products.

The model allows the user to specify either the exploration dollars to be spent or the minimum reserves to be discovered. If the user specifies the dollars to be spent, it can be either the gross expenditures or the expenditures net of any exploration grants. The amount of reserves discovered is then computed using the finding cost applicable to that year. If the user specifies the minimum reserves to be discovered, the model computes the exploration dollars required.

Once the product reserves are determined, the estimated market date for the product is used to schedule the development expenditures. Development costs, expressed as a dollar per unit of product, determine the expenditures required. The total development expenditures are then allocated to the various expenditure tax classes using industry averages.

Production is scheduled by applying production decline curves to the developed reserves and operating costs are calculated from the annual production.

A sample

Figure 2 indicates the data specific to a sample case of exploration in western Canada. All variables not defined in the case assume the default values shown in Figure 1.

Figures 3 through 7 display the output from the sample case. This output is then used by CanadianOxy's financial planning model to compute the resulting economics.

Conclusion

Our purpose in developing the exploration model was to assist the Exploration Department in identifying their assumptions regarding finding costs, development costs, market dates, etc., and to provide them with the economic results of those assumptions immediately. These assumptions can then be compared to the company's own past performance or to the industry's performance to see if they are reasonable.

The development of the model also demonstrated to us, again, the superiority of APL for this type of application. On the basis of a few short meetings with the Exploration Department and subsequent definition of the relationships involved, we sat at the terminal and interactively designed a working model within a week. Through subsequent use the model has been updated and enhanced, and probably will continue to be.

Figure 1
Description Of The Variables Used In
The Model And Their Default Values

[1] STP+1981 # THE STARTING YEAR FOR THE FORECAST,
[2] TPR+10 # THE NUMBER OF YEARS IN THE FORECAST,
[3] IFR+0.10 # THE GENERAL INFLATION RATE USED IN THE FORECAST,
[4] BONUS+ALL 0.3 # THE AMOUNTS SPENT FOR LAND BONUSES (\$ OR PCT OF TOTAL EXPLORATION \$),
[5] EXDRILL+ALL 0.8 # THE AMOUNTS SPENT ON EXPLORATION DRILLING (\$ OR PCT OF TOTAL EXPLORATION \$ MINUS BONUS \$ AND RENT \$),
[6] RENTS+ALL 1000 # THE AMOUNTS SPENT ON LEASE RENTALS (\$ OR PCT OF BONUS \$),
[7] SEISMIC+ALL 0.20 # THE AMOUNTS SPENT ON SEISMIC (\$ OR PCT OF TOTAL EXPLORATION \$ MINUS BONUS \$ AND RENT \$),
[8] ALTADIC+ALL 0 # PROVINCIAL DRILLING INCENTIVE GRANTS (\$ OR PCT OF EXPLORATION DRILLING \$),
[9] PIPEX+ALL 0 # FEDERAL P,I,P, GRANTS FOR EXPLORATION (\$ OR PCT OF SEISMIC AND EXPLORATION DRILLING \$),
[10] TOTEXPL+ALL 0 # TOTAL EXPLORATION DOLLARS (SPECIFIED TOTAL \$ OR DEFAULTS TO SUM OF BONUS,SEISMIC,DRILLING + RENT \$),
[11] NETEXPL+ALL 0 # NET EXPLORATION EXPENDITURES AFTER ALL INCENTIVE GRANTS (SPECIFIED \$ OR RESULT OF TOTEXPL-ALTADIC+PIPEX),
[12] ODA+ALL 0.5 # PROPORTION OF EXPLORATION DOLLARS DIRECTED AT OIL PLAYS (DECIMAL),
[13] OFO+ALL 0.5 # CHANCES OF FINDING OIL WHEN DOLLARS ARE DIRECTED AT OIL PLAYS (DECIMAL),
[14] OFG+ALL 0.2 # CHANCES OF FINDING OIL WHEN DOLLARS ARE DIRECTED AT GAS PLAYS (DECIMAL),
[15] BIFR+SIFR+DIFR+IFR # INFLATION RATES FOR BONUS,SEISMIC + DRILLING RELATIVE TO GENERAL INFLATION RATE,
[16] RIFR+0 # RENTAL INFLATION RATE,
[17] GFIFR+OFIFR+IFR+0.05 # INFLATION RATES FOR GAS AND OIL FINDING COSTS RELATIVE TO GENERAL INFLATION RATE,
[18] GDIFR+ODIFR+IFR # INFLATION RATES FOR GAS AND OIL DEVELOPMENT COSTS RELATED TO GENERAL INFLATION RATE,
[19] GFINDCOST+0.5 INFL GFIFR # GAS FINDING COSTS (\$/MCF INFLATED),
[20] OFINDCOST+2.9 INFL OFIFR # OIL FINDING COSTS (\$/BDL INFLATED),
[21] GDEVCOSt+0.15 INFL GDIFR # GAS DEVELOPMENT COSTS (\$/MCF INFLATED),
[22] ODEVCOSt+0.90 INFL ODIFR # OIL DEVELOPMENT COSTS (\$/BDL INFLATED),
[23] GDEVSHD+.3 .15 .1 .1 .05 # GAS DEVELOPMENT SCHEDULE (PROPORTION OF TOTAL RESERVES DEVELOPED BY YEAR - STARTING THE YEAR BEFORE PRODUCTION),
[24] ODEVSHD+.5 .3 0 0 .2 # OIL DEVELOPMENT SCHEDULE (PROPORTION OF TOTAL RESERVES DEVELOPED BY YEAR - STARTING THE YEAR BEFORE PRODUCTION),
[25] GDRLPCT+ 0.6 # PROPORTION OF GAS DEVELOPMENT COSTS THAT ARE CLASSIFIED AS DRILLING COSTS,
[26] GGATHPCT+ 0.29 # PROPORTION OF GAS DEVELOPMENT COSTS THAT ARE CLASSIFIED AS GATHERING COSTS,
[27] GPLTPCT+ 0.11 # PROPORTION OF GAS DEVELOPMENT COSTS THAT ARE CLASSIFIED AS PLANT COSTS,
[28] ODRLPCT+ 0.75 # PROPORTION OF OIL DEVELOPMENT COSTS THAT ARE CLASSIFIED AS DRILLING COSTS,
[29] OGATHPCT+ 0.25 # PROPORTION OF OIL DEVELOPMENT COSTS THAT ARE CLASSIFIED AS GATHERING COSTS,
[30] GPRODLAY+(6 6 5 4 3 3 3) THEN 2 # GAS PRODUCTION DELAY IN YEARS FROM THE TIME THE RESERVES ARE DISCOVERED,
[31] OPRODLAY+ALL 1 # OIL PRODUCTION DELAY IN YEARS FROM THE TIME THE RESERVES ARE DISCOVERED,
[32] GPRODSHD+.055 10 .1 # GAS PRODUCTION SCHEDULE (1, PCT OF RESERVES 2, NO. OF YRS CONSTANT PRODUCTION 3, ANNUAL DECLINE FACTOR),
[33] OPRODSHD+.06 5 .1 # OIL PRODUCTION SCHEDULE (1, PCT OF RESERVES 2, NO. OF YRS CONSTANT PRODUCTION 3, ANNUAL DECLINE FACTOR),
[34] BOPD+ALL 100 # AVERAGE PRODUCTION FROM EACH OIL WELL IN BARRELS OF OIL PER DAY,
[35] PROD+2 3P' GAS OIL ' # UNIT HEADINGS,
[36] PU+2 2 15P' (\$/MCF) (MCMF) (\$/BDL) (MBDL) ' # UNIT HEADINGS,
[37] PIPDEV+ALL 0 # FEDERAL P,I,P, GRANTS FOR DEVELOPMENT (\$ OR PCT OF TOTAL DEVELOPMENT \$),
[38] TITLE+'CANADIAN OCCIDENTAL PETROLEUM LTD.',
[39] FEDERALBACKIN+ALL 0 # THE PERCENT WORKING INTEREST THAT REVERTS TO THE FEDERAL GOVERNMENT,
[40] OILRES+ALL 0 # THE MINIMUM AMOUNT OF OIL RESERVES YOU WANT TO DISCOVER (MAY BE MORE DEPENDING ON OIL/GAS FINDING RATIOS),
[41] GASRES+ALL 0 # THE MINIMUM AMOUNT OF GAS RESERVES YOU WANT TO DISCOVER (MAY BE MORE - SEE OILRES NOTE),
[42] OEDNMHOIL+ALL 0 # THE OPERATING COST FOR OIL WELLS IN M\$ PER WELL MONTH,
[43] OEDPUOIL+ALL 0 # THE OPERATING COST FOR OIL PRODUCTION IN \$ PER BDL,
[44] OEDPUGAS+ALL 0 # THE OPERATING COST FOR GAS PRODUCTION IN \$ PER MCF,
[45] 9

Figure 2
Input For Sample Case Showing
Exploration In Western Canada

```

KEY VARS; TP  IFR
[1]  STP+1983
[2]  IFR+.091 .081 .075 .07 .07 THEN .06
[3]  TOTEXPL+17685.5 19679.7 21740.4 23857.4 26114.3 28247 30450.7 32697.2 34947.1 37146.8
[4]  ALTADIC+ (3 OF 0.15) THEN 0
[5]  RENTS+ALL 900
[6]  ODA+ALL 1.0
[7]  OFO+ALL 0.6
[8]  GFINDCOST+0.40 INFL GFIFR
[9]  OFINDCOST+2.36 INFL OFIFR
[10] GDEVCOST+0.30 INFL GDIFR
[11] ODEVCOST+1.77 INFL ODIFR
[12] OPRODSHD+0.06 10 0.12
[13] NETEXPL+ALL 0
[14] PIPDEV+ALL 0
[15] PIPEX+ALL 0
[16] CASETITLE+'WESTERN CANADA EXPLORATION PROGRAM'
[17] OEDPUGAS+0.28 INFL IFR
[18] OEDPUOIL+2.00 INFL IFR
WITH SYSTEM DATED 1982/05/17 15:17:45

```

CALC

NOTE; THE PRODUCTION SCHEDULING PARAMETERS 0.055 10 0.1 PRODUCE 94.3 PCT OF THE RESERVES,

NOTE; THE PRODUCTION SCHEDULING PARAMETERS 0.06 10 0.12 PRODUCE 97.5 PCT OF THE RESERVES,

...,CALCULATIONS DONE,

TABLE 1 2 3 4 5
ALIGN PAPER,....

CURRENT CASE:1983-TEN/YR-WESTERN
 ***WESTERN CANADA EXPLORATION PROGRAM

CANADIAN OCCIDENTAL PETROLEUM LTD.,
 GROSS EXPLORATION EXPENDITURE FORECAST

('000\$)

YEAR	LEASE ACQUISITION COSTS	LEASE RENTALS	SEISMIC COSTS	EXPLORATION DRILLING COSTS	TOTAL SEISMIC + DRILLING COSTS	GROSS EXPLORATION EXPENDITURE	EXPLORATION INCENTIVE GRANTS	NET EXPLORATION EXPENDITURE
1983	5306	900	2296	9184	11480	17685	(1378)	16308
1984	5904	900	2575	10301	12876	19680	(1545)	18135
1985	6522	900	2864	11455	14318	21740	(1718)	20022
1986	7157	900	3160	12640	15800	23857	0	23857
1987	7834	900	3476	13904	17380	26114	0	26114
1988	8474	900	3775	15098	18873	28247	0	28247
1989	9135	900	4083	16332	20415	30451	0	30451
1990	9809	900	4398	17590	21988	32697	0	32697
1991	10484	900	4713	18850	23563	34947	0	34947
1992	11144	900	5021	20082	25103	37147	0	37147
TOTALS	81770	9000	36359	145437	181796	272566	(4641)	267925
	=====	=====	=====	=====	=====	=====	=====	=====

EXPLORATION ASSUMPTIONS:

LEASE ACQUISITION COSTS - INFLATED 9.1 8.1 7.5 7.0 7.0 6.0 6.0 6.0 6.0 6.0 PCT,
 LEASE RENTALS - INFLATED 0.0 PCT,
 SEISMIC COSTS - INFLATED 9.1 8.1 7.5 7.0 7.0 6.0 6.0 6.0 6.0 6.0 PCT,
 DRILLING COSTS - INFLATED 9.1 8.1 7.5 7.0 7.0 6.0 6.0 6.0 6.0 6.0 PCT,

DRILLING INCENTIVE CREDITS - 15 PCT, OF ALBERTA DRILLING COSTS - 25 PCT OF P,I,P, ELIGIBLE SEISMIC + DRILLING COSTS

EXPENDITURE DIRECTED TO GAS EXPLORATION - 0.0 PCT & EXPENDITURE RESULTING IN GAS FINDS - 40.0 PCT

GAS FINDING COST - 0.40 \$/MCF IN 1982 - INFLATED 14.1 13.1 12.5 12.0 12.0 11.0 11.0 11.0 11.0 11.0 PCT PER YR

EXPENDITURE DIRECTED TO OIL EXPLORATION - 100.0 PCT & EXPENDITURE RESULTING IN OIL FINDS - 60.0 PCT

OIL FINDING COST - 2.36 \$/BBL IN 1982 - INFLATED 14.1 13.1 12.5 12.0 12.0 11.0 11.0 11.0 11.0 11.0 PCT PER YR

Gross Exploration Expenditure Forecast

Figure 3

Figure 4

Exploration Allocation And Discovered Reserves Forecast

CANADIAN OCCIDENTAL PETROLEUM LTD., EXPLORATION ALLOCATION AND DISCOVERED RESERVES FORECAST									

('000\$)									

* GAS *									

YEAR	LEASE ACQUISITION COSTS	LEASE RENTALS	SEISMIC COSTS	EXPLORATION DRILLING COSTS	TOTAL SEISMIC + DRILLING COSTS	EXPLORATION EXPENDITURE ALLOCATED TO GAS	GAS FINDING COST (\$/MCF)	GAS RESERVES DISCOVERED (MMCF)	NET CUMULATIVE RESERVES (MMCF)
----	-----	-----	-----	-----	-----	-----	-----	-----	-----
1983	2122	360	918	3674	4592	7074	0.46	15500	15500
1984	2362	360	1030	4120	5150	7872	0.52	15250	30750
1985	2609	360	1145	4582	5727	8696	0.58	14975	45725
1986	2863	360	1264	5056	6320	9543	0.65	14673	60398
1987	3134	360	1390	5562	6952	10446	0.73	14340	74737
1988	3390	360	1510	6039	7549	11299	0.81	13974	88711
1989	3654	360	1633	6533	8166	12180	0.90	13571	101430
1990	3924	360	1759	7036	8795	13079	1.00	13128	110447
1991	4194	360	1885	7540	9425	13979	1.11	12641	118209
1992	4458	360	2008	8033	10041	14859	1.23	12105	123967
TOTALS	32708	3600	14544	58175	72719	109026		140156	
	=====	=====	=====	=====	=====	=====		=====	

* OIL *									

YEAR	LEASE ACQUISITION COSTS	LEASE RENTALS	SEISMIC COSTS	EXPLORATION DRILLING COSTS	TOTAL SEISMIC + DRILLING COSTS	EXPLORATION EXPENDITURE ALLOCATED TO OIL	OIL FINDING COST (\$/BBL)	OIL RESERVES DISCOVERED (MMBL)	NET CUMULATIVE RESERVES (MMBL)
----	-----	-----	-----	-----	-----	-----	-----	-----	-----
1983	3183	540	1378	5510	6888	10611	2.69	3941	3941
1984	3542	540	1545	6180	7725	11808	3.05	3877	7581
1985	3913	540	1718	6873	8591	13044	3.43	3807	10919
1986	4294	540	1896	7584	9480	14314	3.84	3730	13952
1987	4701	540	2086	8342	10428	15669	4.30	3646	16677
1988	5084	540	2265	9059	11324	16948	4.77	3553	19089
1989	5481	540	2450	9799	12249	18270	5.30	3450	21186
1990	5885	540	2639	10554	13193	19618	5.88	3338	22964
1991	6290	540	2828	11310	14138	20968	6.52	3214	24417
1992	6686	540	3012	12049	15062	22288	7.24	3078	25541
TOTALS	49062	5400	21816	87262	109078	163540		35633	
	=====	=====	=====	=====	=====	=====		=====	

CANADIAN OCCIDENTAL PETROLEUM LTD.
DEVELOPMENT EXPENDITURE FORECAST

('000\$)

YEAR	GAS DEVELOPMENT					OIL DEVELOPMENT				DEVELOPMENT	DEVELOPMENT
	DRILLING	GATHERING	PLANT	TOTAL	\$/MCF	DRILLING	GATHERING	TOTAL	\$/BBL	GRAND TOTAL	INCENTIVE GRANTS
1983	0	0	0	0	0.33	2854	951	3805	1.93	3805	0
1984	0	0	0	0	0.35	4886	1629	6515	2.09	6515	0
1985	0	0	0	0	0.38	5161	1720	6882	2.24	6882	0
1986	0	0	0	0	0.41	5416	1805	7221	2.40	7221	0
1987	0	0	0	0	0.44	7188	2396	9583	2.57	9583	0
1988	1288	622	236	2146	0.46	7446	2482	9928	2.72	12074	0
1989	6582	3181	1207	10970	0.49	7691	2564	10255	2.89	21225	0
1990	7558	3653	1386	12597	0.52	7918	2639	10557	3.06	23154	0
1991	7467	3609	1369	12445	0.55	8119	2706	10825	3.24	23270	0
1992	7473	3612	1370	12454	0.58	8286	2762	11048	3.44	23503	0
1993	7236	3498	1327	12061	0.62	4410	1470	5880	3.64	17940	0
1994	4933	2384	904	8222	0.65	1934	645	2579	3.86	10801	0
1995	2685	1298	492	4476	0.69	1974	658	2632	4.09	7108	0
1996	1682	813	308	2803	0.74	2004	668	2672	4.34	5474	0
1997	862	417	158	1437	0.78	0	0	0		1437	0
1998	300	145	55	500	0.83	0	0	0		500	0
TOTALS	48067	23232	8812	80111		75286	25095	100381		180492	0
	=====	=====	=====	=====		=====	=====	=====		=====	=====

DEVELOPMENT ASSUMPTIONS:

GAS

DEVELOPMENT DELAY - PRODUCTION DELAY LESS ONE YR
DEVELOPMENT SCHEDULE - 30 PCT IN YR 1; 30 PCT IN YR 2; 15 PCT IN YR 3; 10 PCT IN YR 4; 10 PCT IN YR 5; 5 PCT IN YR 6
EXPENDITURE ALLOCATION - 60 PCT DRILLING; 29 PCT GATHERING; 11 PCT PLANT
DEVELOPMENT COST - 0.30 \$/MCF IN 1982 - INFLATED 9.1 8.1 7.5 7.0 7.0 6.0 6.0 6.0 6.0 6.0 PCT PER YR

OIL

DEVELOPMENT DELAY - PRODUCTION DELAY LESS ONE YR
DEVELOPMENT SCHEDULE - 50 PCT IN YR 1; 30 PCT IN YR 2; 0 PCT IN YR 3; 0 PCT IN YR 4; 20 PCT IN YR 5
EXPENDITURE ALLOCATION - 75 PCT DRILLING; 25 PCT GATHERING
DEVELOPMENT COST - 1.77 \$/BBL IN 1982 - INFLATED 9.1 8.1 7.5 7.0 7.0 6.0 6.0 6.0 6.0 6.0 PCT PER YR

Development Expenditure Forecast

Figure 5

CANADIAN OCCIDENTAL PETROLEUM LTD.,
PRODUCTION FORECAST

YEAR	GAS PRODUCTION (MMCF)	OIL PRODUCTION (MBBL5)	NO. OF OIL WELLS
1983	0	0	
1984	0	236	6
1985	0	469	13
1986	0	698	19
1987	0	921	25
1988	0	1140	31
1989	853	1353	37
1990	4111	1560	43
1991	4879	1760	48
1992	6348	1953	54
1993	7043	2138	59
1994	7709	2110	59
1995	7709	2057	59
1996	7709	1983	59
1997	7709	1891	59
1998	7709	1784	59
1999	7623	1664	59
2000	7221	1533	59
2001	6782	1395	59
2002	6240	1250	59
2003	5682	1100	59
2004	5114	968	59
2005	4603	852	59
2006	4142	749	59
2007	3728	659	59
2008	3355	580	59
2009	3020	480	52
2010	2718	392	46
2011	2446	316	39
2012	2201	249	33
2013	1981	191	27
2014	1625	140	22
2015	859	97	16
2016	631	59	10
2017	295	27	5
2018	137	0	
TOTALS	132179	34754	

PRODUCTION ASSUMPTIONS:

GAS

 PRODUCTION DELAYS - 6 6 5 4 3 3 3 2 2 2 YRS
 INITIAL PRODUCTION - 5.5 PCT OF RESERVES
 FLAT PRODUCTION PERIOD - 10 YRS
 PRODUCTION DECLINE - 10 PCT PER YR
 PRODUCTION LIFE - 25 YRS

OIL

 PRODUCTION DELAYS - 1 1 1 1 1 1 1 1 1 1 YRS
 INITIAL PRODUCTION - 6.0 PCT OF RESERVES
 FLAT PRODUCTION PERIOD - 10 YRS
 PRODUCTION DECLINE - 12 PCT PER YR
 PRODUCTION LIFE - 25 YRS
 AVGE, PRODUCTION PER WELL - 100 BOPD

Production Forecast

Figure 6

Figure 7

Operating Expenses

CANADIAN OCCIDENTAL PETROLEUM LTD. OPERATING EXPENSES			

('000\$)			
	GAS OPERATING EXPENSE -----	OIL OPERATING EXPENSE -----	TOTAL OPERATING EXPENSE -----
1983	0	0	0
1984	0	558	558
1985	0	1189	1189
1986	0	1892	1892
1987	0	2675	2675
1988	0	3508	3508
1989	389	4414	4803
1990	1990	5395	7384
1991	2504	6452	8956
1992	3452	7589	11041
1993	4060	8804	12865
1994	4711	9209	13920
1995	4994	9517	14510
1996	5293	9725	15018
1997	5611	9830	15441
1998	5947	9830	15777
1999	6234	9719	15954
2000	6260	9495	15755
2001	6232	9155	15386
2002	6078	8694	14771
2003	5867	8109	13976
2004	5597	7564	13161
2005	5339	7056	12396
2006	5094	6582	11676
2007	4859	6140	10999
2008	4636	5727	10363
2009	4423	5022	9445
2010	4219	4351	8570
2011	4025	3711	7737
2012	3840	3101	6941
2013	3663	2519	6183
2014	3185	1964	5149
2015	1784	1435	3219
2016	1389	931	2320
2017	690	453	1142
2018	339	0	339

OPERATING EXPENSE ASSUMPTIONS:

GAS

OPERATING COST - 0.28 \$/MCF IN 1982

INFLATED 9.1 8.1 7.5 7.0 7.0 6.0 6.0 6.0 6.0 6.0 PCT PER YR

OIL

OPERATING COST - 2.00 \$/BBL AND 0 \$/WELL-MONTH IN 1982

INFLATED 9.1 8.1 7.5 7.0 7.0 6.0 6.0 6.0 6.0 6.0 PCT PER YR

COUNTRY RISK ANALYSIS: A QUANTITATIVE APPROACH TO THE BANK OF BRITISH COLUMBIA

Christopher Dobrzanski
Economist, International Analyses
Bank of British Columbia
Vancouver, British Columbia

Summary

Country evaluation and risk assessment are integral parts of the Bank of British Columbia's portfolio management process. The global environment in which the Bank operates is constantly changing. Risks and opportunities alike must be identified sufficiently in advance to allow management to adjust the portfolio to maximize shareholders' net worth.

Country risk encompasses the whole spectrum of risks that arise from the economic, social, legal, and political conditions of a foreign country. These factors may have potentially favourable or adverse consequences for loans to borrowers in a country. To assess a country's economic strength and debt structure, a series of economic indicators are used with country risk analysis.

The financial strength is examined by using a liquidity index to measure the cash flow position of the country. Unlike the debt service ratio that focuses on the ability to meet interest and principal payments, the liquidity measure offers some indication if the borrowers will be forced to finance deficits by foreign debt.

Independent evaluation of the countries is brought together with an analysis of consolidated chartered bank lending. Analysis of exposure levels and concentrations, based on U.S. and Canadian aggregate data, is a benchmark for management strategy. Diversification remains a bank's best protection against risk in an uncertain world.

Public and private data banks are identified and sample analyses based on APL applications are provided. They address the risk of the country over time and the risk to the portfolio. The quantitative approach is required to maintain adequate internal mechanisms for monitoring and controlling country exposure.

Status of management information needs

Commercial banks have been developing country risk analyses systems in earnest since the 1970's. The rapid expansion of Eurocurrency markets bolstered by the U.S. Foreign Credit Restraint Program in 1968-69 and recycling of OPEC surpluses in 1973-74

created an abundant supply of loanable funds. Demand for Eurocurrency loans was also growing from U.S. Corporations and non-oil producing countries. This unexpected shift in resources via the financial intermediation of commercial banks created new information needs for banks:

- (i) How to design a portfolio of international assets that would be consistent with conventional portfolio concepts of minimizing risk and maximizing return.
- (ii) How to identify a borrower's current and future ability to repay interest and principal in view of the whole spectrum of risks that arise from the economic, social, legal, and political conditions of a foreign country.
- (iii) How to assess pricing terms on foreign loans with respect to the underlying risk factors of a particular loan maturity and to predict future loan terms.

Data bases have been the prerequisite to answer the portfolio design, country risk, and loan yields for bank management. Country data in machine readable form was available from the International Monetary Fund and the World Bank; hence, credit-worthiness has been the first area to be researched.

Portfolio structure of Canadian chartered banks is confidential whereas it is published for West Germany and the United States. In Canada, the Office of the Inspector General of Banks in the Ministry of Finance has been gathering the country-by-country data since June 1978. The current Bank Act revisions make the data base more comprehensive as at November 30th, 1981 since they now include loans made by corporations in which the banks own 20 or more percent of the voting stock. The data is returned to the Chief Accountant of each Schedule A Bank on a total bank/own bank basis.

Data on terms and maturities of borrowings are scarce. World Bank data are published only on publicized Eurocurrency credits. A complete risk-return framework cannot be completed until the already documented risk analysis can be correlated with the data on terms and maturities of foreign debt.

Techniques for country risk assessment

International lending involves country risk. Specific risks in country analysis include political or social upheaval, nationalization or expropriation, government repatriation of external debt, exchange controls, or foreign exchange shortfalls that might preclude a country from complying with the terms of its international loans.

A country's economic situation and debt structure are the areas most commonly quantified using the IFS and World Debt data tables. While recognising that there is no adequate set of ratios or formulae that would consider all relevant variables, there is some consensus on a core subset.

The ratios that have regularly been associated with the incidence of countries facing financial distress can be grouped under seven headings:

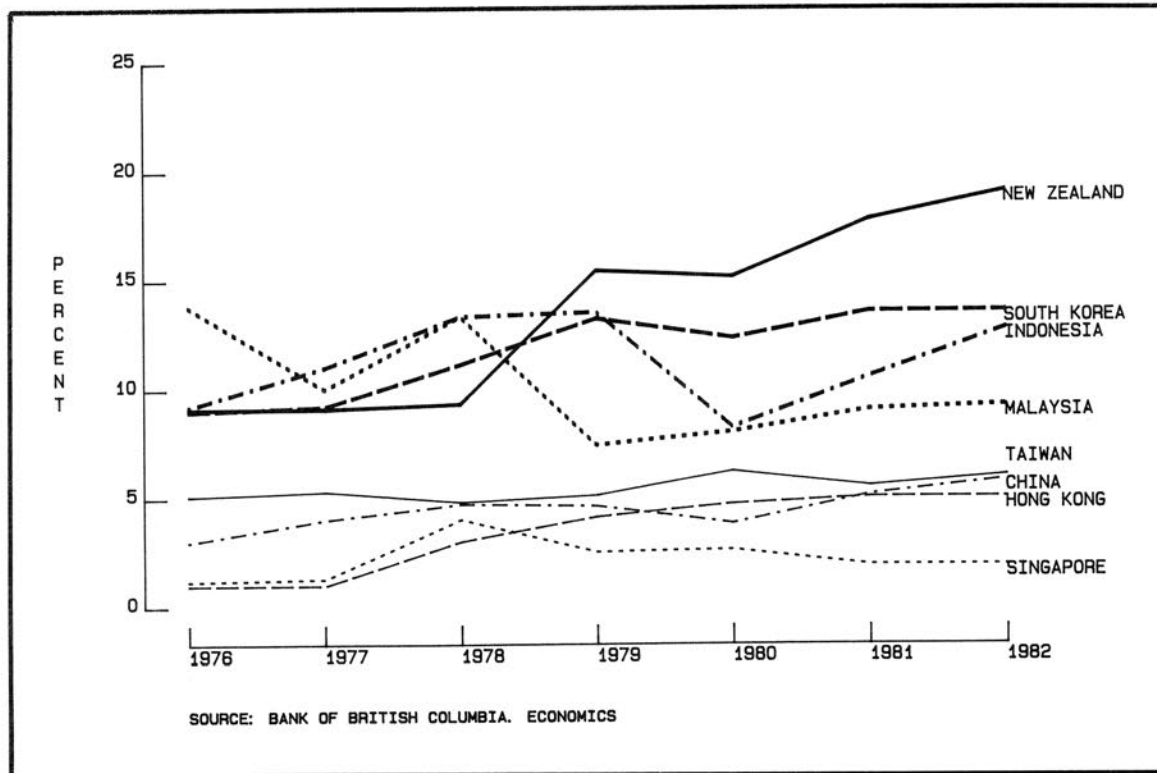
- 1) ECONOMIC DEVELOPMENT: GNP Level and Growth, Investment Relative Share and Growth, Inflation Rate, Money Supply Creation, National Government Budget.
- 2) IMPORT AND EXPORT STRUCTURE: Food and/or Energy Share of Imports, Commodity Concentration of Exports, Geographical Diversification of Exports.
- 3) BALANCE OF PAYMENTS: Current Account, Trade Balance, Service Balance, Capital Flows, all as a share of GNP.
- 4) CURRENCY STRENGTH AND STABILITY: Comparative Purchase Parity, Effective Exchange Rate Index.
- 5) INTERNATIONAL LIQUIDITY: Months of Import Coverage, Gross International Reserves Excluding and Including Gold Reserves.
- 6) SERVICE ACCOUNT: Transfer of Dividend and Interest Payments Share of Service Account, Tourism and Transport as a Share of Service Account.
- 7) DEBT SERVICE: Debt Service Ratio, Foreign Debt Share of GNP, Average Interest and Principal Payments as a Percent of Export Earnings.

These ratios, together with political indicators, would be used to identify current conditions and likely direction of economic activity. Because the data bases can be accessed by computer terminals, the economic status of a country can be readily quantified.

Among the ratios, the debt service ratio which is defined as total interest and principal payments divided by export earnings from goods and services is most frequently cited. The debt service ratio is directed directly to the measurement of a country's ability to service debt. Its level of 30% has developed into a rule-of-thumb maximum without a theoretical underpinning.

Figure 1 illustrates the debt service ratio of South East Asian Economies. The results correlate poorly with other rankings of overall country risk. They do not agree either with the financial market's perception which continues to give New Zealand and Malaysia finer terms than South Korea or Taiwan. The Debt Service Ratio, therefore, must be evaluated in conjunction with other economic variables.

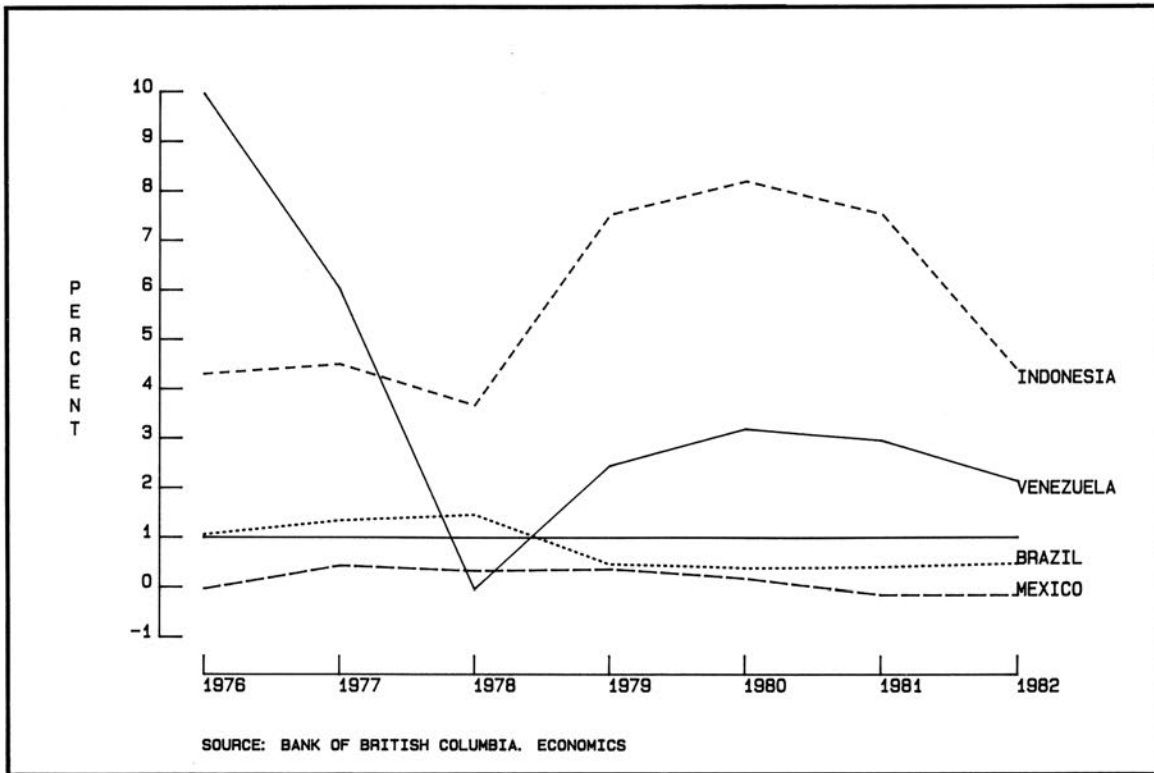
Figure 1
Asia Debt Service Ratio
in Perspective



Another dimension to the debt burden and debt capacity of a country is its cash flow. A country's cash flow is defined as foreign exchange earnings plus foreign exchange reserves and loans committed but not drawn down, expressed as a ratio relative to debt service payments for one year ahead. If the ratio falls below 1.0 then the country is dependent on new borrowings to service the debt.

As illustrated in Figure 2 both Brazil and Mexico have a liquidity index below 1.0. They also have a spread over the London Interbank Offered Rate (LIBOR) that is significantly above their norms (either geographic or economic groupings).

Figure 2
Country Risk Analysis
Cashflow Ratio-Liquidity Measure
(Below 1 Percent Indicates a Liquidity Problem)



These indices have been obtained using MAGIC to get IFS and World Debt Tables. It is recommended that the commands be set up as a function that can execute the seven groups of economic ratios and the liquidity index. Country codes for IFS accounts do vary, so some editing will be required. The flexibility of MAGIC and APL allows the country risk analyst to adapt to changing economic concerns with minimal frustration.

The quantitative approach to assess economic conditions merely frees up time to focus on the unique country aspects that defy a two-dimensional presentation. Given the range of factors that can affect projections based on past statistical regularity, a descriptive and analytical country case study has some degree of forecast error. To minimize potential losses, diversification remains a bank's best protection against risk in an uncertain world.

Portfolio structure of Canadian chartered banks

The data on Canadian bank lending to foreign countries and corporations is confidential. The Bank of Canada publishes details on the assets booked in Canada to the United States and the United Kingdom. Some chartered banks reveal their geographical distribution by country bloc or country per capita income in their annual reports. Although the Inspector General of Banks in the Finance Ministry makes the data available to each chartered bank, the confidential nature of data will be respected.

Data received by Canadian chartered banks from the Quarterly Geographical Distribution of Assets and Liabilities Return can be efficiently analyzed using WIZARD. This APL program is a very useful multidimensional data base management system that can handle the detail provided (over 130 countries, 5 currencies, and 90 asset and liability categories).

Country exposure data is constructed by examining non-local claims and liabilities booked at head office and Canadian branches, non-offshore foreign branches, agencies and consolidated subsidiaries; and offshore foreign branches, agencies, and consolidated subsidiaries.

This data is a prerequisite to identify what is an efficient portfolio. A portfolio has two types of risk: unsystematic (diversifiable) risk, and systematic (non-diversifiable) risk. The loan portfolio risk can be reduced if the unsystematic risk is reduced. Reducing the share in a high risk country or increasing the share in a non-correlated country would lower the unsystematic risk in a portfolio. The value of diversification, however, is limited by the presence of systematic risk (i.e., events that affect many countries simultaneously such as reduced export opportunities).

Unlike country-by-country analysis of country risk assessment, where research has been exchanged freely due to the public nature of the basic data, portfolio performance is kept confidential. Regulatory bodies do not disclose their ratings since they have been monitoring rather than restricting lending by U.S. and Canadian banks. Consolidated Canadian chartered bank foreign lending, nevertheless, is a benchmark for portfolio performance.

To illustrate the nature of portfolio analysis at an aggregate level, published estimates of international exposure are used rather than actual figures. The estimates illustrate the rapid changes in the portfolio preferences. Areas with positive economic and political prospects have improved their relative share of international assets.

In Canada, Schedule A Banks can create a country-by-country portfolio analysis using the quarterly data tape. With the availability of the new consolidated data on Canadian banks' geographical distribution of international assets and the use of WIZARD, the analyst can monitor changes in:

- (i) outstanding and relative share of each country
- (ii) maturity distribution of assets and liabilities
- (iii) maximum exposure including contingent claim.

Figure 3
Canadian Multinational Banks
Estimated Geographical Distribution of International Assets
1980

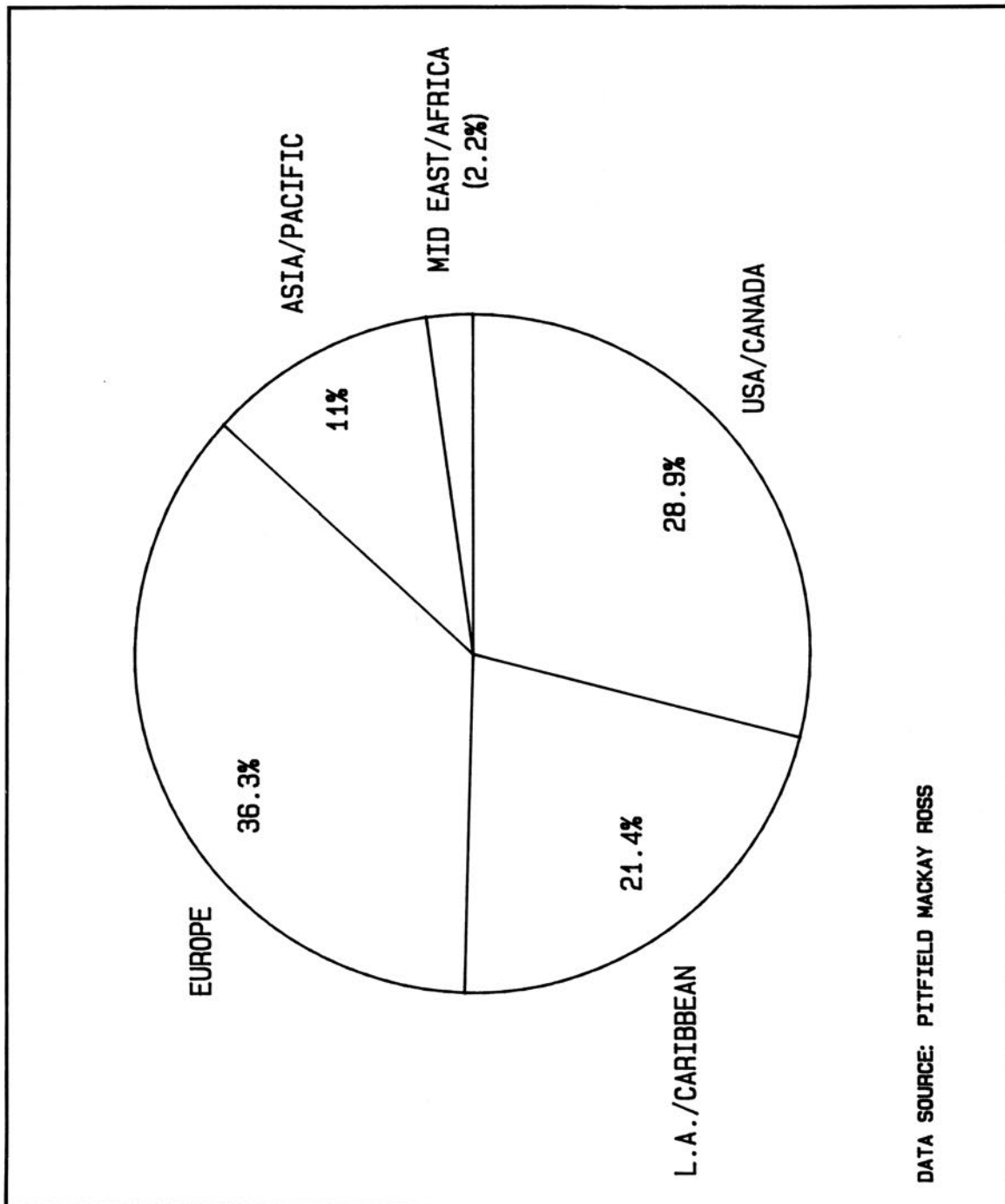


Figure 4
Canadian Multinational Banks
Estimated Geographical Distribution of International Assets
1981

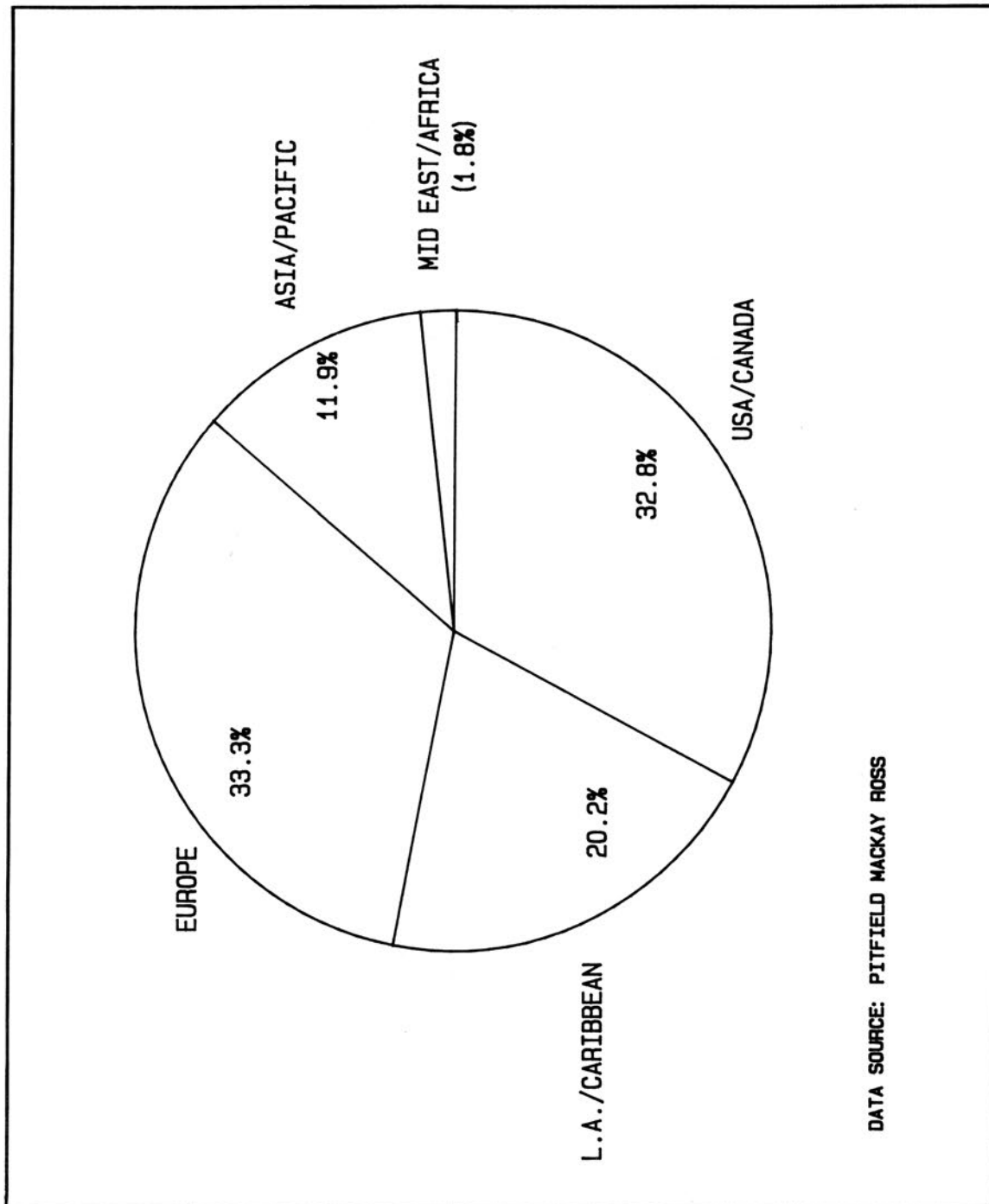
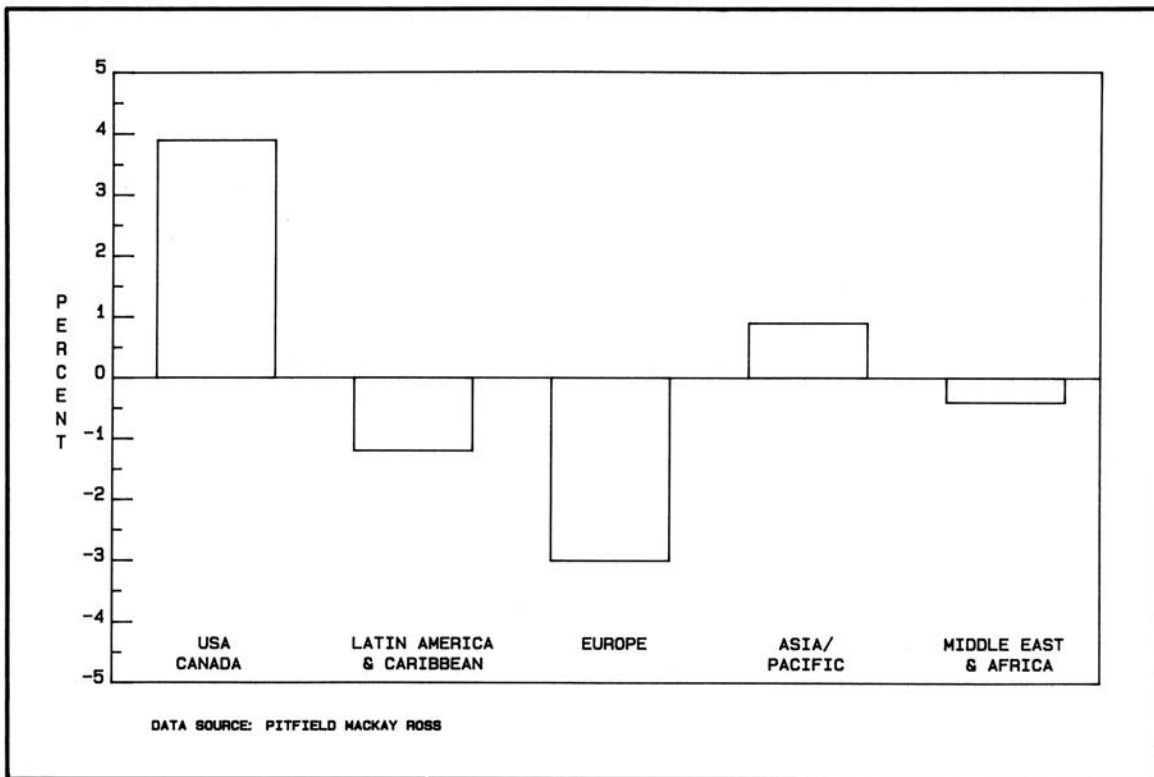


Figure 5
Canadian Multinational Banks
Estimated Geographical Distribution of International Assets
Change From 1980 to 1981



Risk-return of international portfolio

To complete the traditional portfolio analysis, a vector of yields has to be correlated with the country asset data. No such data base exists. An area for future data base builders would include:

- (i) the spread over cost of funds (usually the London Interbank Offered Rate, LIBOR)
- (ii) the management and participation fees of lead managers, co-managers and agents
- (iii) the maturity period and years of grace

The expected return would be calculated as the present value of the above.

Until such data bases exist, the country risk analyst will be limited to elements of risk (country-by-country assessment of political and economic conditions), elements of return (current pricing terms and maturities), and elements of the portfolio (current country-by-country exposure of international assets). The complete picture for aggregate portfolio planning is constrained.

Regardless of data constraints, the quantitative approach is required to maintain adequate internal mechanisms for monitoring and controlling country exposure. Diversification, hence portfolio analysis, remains a bank's best protection against risk in an uncertain world.

COMPREHENSIVE CORPORATE BUDGETING—PLANNING, TRACKING AND COMMITMENT CONTROL WITH APL

**C. Alan Ford
C.A. Ford Management Ltd.
Toronto, Ontario**

Introduction

Budgeting is a much-unloved subject. It is as much unclearly understood. The scope is wider than many managers realize. What is clear is the cognitive dissonance over the usual demand for meticulous treatment of numbers which, being drawn from a usually-misty crystal ball, inherently lack precision. And budgeting is disciplinary, a stick seeming to threaten the spokes of one's operational plans or career goals. Objectives of this paper are the alleviation of paranoia about budgeting and the enhancement of affection for APL. Love budgets, though? Forget it.

APL has characteristics which make it a natural alleviant at least to the drudgery of arithmetic precision. Cross-totals...foot-totals...grand totals? never again does one spend precision time chasing numerical problems when balancing. The operations $+/$, $+/[]$ and $+/+/$ put an end to these problems, thereby engendering affection.

This paper intends, however, to help you know the principal features of comprehensive budgeting. Better the enemy you know than the one you don't. In fact, any line manager should come to know and respect the budget as an essential ally and even friend, flaky numbers and all. In this manner may paranoia be exorcised.

Overview

Budgeting has three interlocking aspects: planning for the future; controlling in the present; analyzing the past to help clarify the crystal ball. Any set of procedures which does not coordinate these three aspects is defective and probably futile.

Corporations may be thorough, lax, or mixed in the treatment of budgeting. Laxness is virtually synonymous with ineffectiveness. Unfortunately, thoroughness alone does not guarantee effectiveness. The other ingredients are the holding of complete and coherent concepts by general management, awareness and support of line management, a helpful modicum of supporting staff, and some useful data processing tools. Toughness then can be effective, a condition worthy of the term "tough-mindedness".

Objectives of budgeting

Budgeting has several essential goals: to force careful planning of future activities in consideration of financial capabilities; to measure the capabilities of line management to plan for the future; to predict cash flow expectations; to enable executive action by line management within boundaries of plan and propriety; to adapt to how events actually unfold in comparison to the operational plans upon which were based the corporate financial plan; to evaluate the performance of line managers in action.

A problem is by definition a variance between a plan and an actuality about which someone of importance cares. Budgets are the plan. Events yield the actualities. Tough-minded management will care about differences and will respond meaningfully when “best-laid plans” go astray, by minimizing the variance subject to essential constraints.

Planning

Planning a budget is only one phase on the plan/control/evaluate loop but it does seem the natural entry point. So let us begin with these assumptions:

- now is the time to prepare next year’s budget plan
- your management unit exists, with a history and a mandate to fulfil
- the history includes a year or more of budget-management records
- the mandate is expressed as a set of expectations—sales levels, production volumes, new project plans—which are meaningful and vital within the larger plans of your company
- you at least are tough-minded, expecting to propose a good, productive, livable budget to which you will manage during the coming year
- you have some workable budget-processing tools at hand

If now *isn’t* budget time, it *is* time to check what of the above you lack. No matter what dictates emerge when budget time arrives, the lack of any of the above at least will make for excessively hard work or at worst make for another exercise in futility. Specifically, and repetitively:

- your unit must have a comprehensive and comprehensible business plan
- it must have constitutional authority to execute the plan
- there must be some quantity of historical data to build on
- you must be ready and able
- your tools must be built and ready

If anything is lacking, quietly but tough-mindedly set about filling the voids *now*.

When budget planning time arrives, the issues become clear:

- What accounts will be active on behalf of your business plans? The accountants have numbers and names for a stack of them. Have they just the ones for you? They can supply new ones for you.
- What accounts are active already? What commitments have been made against them, casting shadows into the new year? Will any new accounts arise to cope with carry-forwards of long range plans?
- What has been the experience and trends on the looser, harder-to-estimate line-items? Do they relate well to other larger or better-known items? Or to measures of business level?

Here are some tips. Recognize items as *soft* or *solid* as to variability, *continuing* or *spotty* as to recurrence of the costs, and *significant* or *minor* in proportion to the whole budget. Treat items in due proportion—solid significant items need more care than soft minor ones. Continuing ones, especially if minor, usually can be estimated as a proportion of a significant continuing item. Watch out for spotty items—once-off purchases may initiate a continuing cost such as monthly maintenance.

Now for the helpful tools. The first is APL alone, in terminal execution mode. I capture in a workspace two APL variables for every line item as I work up my estimates. A character matrix carries brief descriptions, one per row, of every identifiable component under a given account. A numeric matrix carries the monthly estimates for each of those detailed items in rows to match the descriptor. In many instances the detailed numbers for a given row of a specific account can be computed directly from some other line or lines, possibly even deriving from other related account matrices. This level of detail is bedrock to any budget plan (even if not captured in APL variables). The need will be explored further under the Control and Evaluation headings.

Would that all budget-planners were fluent with just a few basics of APL. Direct keyboard action at this stage is so very helpful. The APL programmer working in support of budget-planners, however, can have productive fun building user-helpful routines for the personal use of the budgeteer. At worst, however, this can be a stage of think, pencil, paper and handoff for computer entry and report-back by support staff. Better far that the budgeteer do some on-line thinking, after collecting the historical reports of continuing or functional dependencies. Then the temptation to waste time on manual cross-and-foot-totals will be instantly obviated by means of the APL add/compress/across-or-down operations.

Coping with detail is the issue to this point. Lack of detailed effort by the cognizant managers means they themselves will be dealing with data, not information, with little hope of making good judgement calls or meaningful evaluations in later phases. Moreover, budgets imposed top down or by planning experts cause mental dissociation with the whole process, an even worse condition than arises from lack of detailed thinking.

But there may be many layers of management above the detailers' levels. As the organization tree is ascended each manager's budget becomes increasingly a set of rollups of lower levels of details, with increasingly little own-level detail. Consolidation tools such as those contained in CONSOL are therefore necessary. Consolidations along chains of command and across accounting entities are required, and APL has proven to be well-adapted to those tasks.

What the data processing tools cannot themselves do is convince the higher level manager of the “goodness” of the plan. A word or two needs be devoted to this issue. The challenges are:

- How good are your details?
- How clever are your plans of action, in terms of results expected against costs?
- How well have your subordinates done their planning?

The responses may be reassuring in detail or in general. The detailed responses are myriad, very case-dependent. But there is a general test available, the year-to-year comparison. Provided that there exists a reasonable continuity of function and scale of effort, a figure of merit can be computed as follows:

- a) What is the percentage change in work load, year-to-year?
- b) What is the percentage price change in the dominating cost factor?
- c) What is the percentage change in your budget, year-to-year?
- d) Merit is given by [(a) plus (b) less (c)].

Budget accommodations

There is, in these times, a virtual certainty that a budget planned is a budget to be cut, regardless of merit. Best if such accommodations can be made on the basis of plans to be foregone rather than the blanket order to “cut everything by x percent”. How easy it would be just to scale down the APL data matrices. However, a better plan is to make the APL tools support “budget savings target” detailed lines embedded where most reasonable into the details, if adequate explicit reductions cannot be planned. These “negative contingencies” can become excellent indicators as the budget year progresses.

The approved budget

Finally, however, an officially sanctioned budget should be created. The APL tools will have rolled up the line item details and consolidated them, as finally approved, into successive summary levels ready for the control of action as the year unfolds.

Controlling

The control processes have many aspects, or at least a set of appearances according to one’s vantage point. Lower ranks of management feel heavily controlled. Middle management feels control inputs from above, the difficulties of exerting control upon subordinates, plus some sense of elbow room in playing off Tom’s overrun against Dick’s underrun. And Harry, the President, has still more short-term power to play off the overruns and underruns, or even to make exceptions. “Contingency funds are mine”, sayeth many seniormost executives. That is not inappropriate, since they answer for the overall result, profit or loss.

Here lies a key to the paranoid reaction to budgets at all management levels: at the top, the least control of detail but clearest vision of consequences; at bottom the realities of detail with the least clear comprehension of impact; in the middle the worst of both problems.

APL can help alleviate these paranoid stresses in one way only, by making it easy to know the facts as clearly as numbers can provide such knowledge. But no software or numbers will defend anyone in a situation where the very concept of control is misunderstood. Let us look at control in the systematic input-process-output sense.

The active control process

At any manager's level the primary *input* is the demand events place on his resources, for which read budget allotments. The primary *process* is to make an appropriate management decision.

The *outputs* are *productive results* and *budget impacts*. Ideally, the productive results and the budget impacts are as planned. Exactly! More result with less impact than expected, which would seem even better, actually indicates that planning in this area was not tight enough. The implied result is that something else was forced out of the plans needlessly.

Tracking

Exactly? Nothing will be, except by chance and not for long. As far as budget impacts are concerned, two smoothing processes are available. How have expenditures been, compared to plan, on a year-to-date basis (or over a moving window of, say, the last six months)? And how have the impacts been across the arrays of line items for which one is accountable? Given easy access to the appropriate variables, the APL-using managers can manipulate the data with either sharp or broad focus, supporting the monthly evaluations which tough-minded management knows to be necessary to hold effective attention to secondary control processes.

The variances, differences between budget and actual, are the inputs driving the secondary but vital process of responding to the departure from plan. The outputs reflect any of a variety of decisions: ignore a trivial variance for operating purposes; recognize and correct faulty control of expenditures; recognize and adapt to external factors or loads in variance from plan; divert resources from underspent to overspent accounts where appropriate.

All this while remembering that *what* is being controlled is the performance of an organization and that the managers are those *who* exert control. And that both productive results and costs are factors to be controlled. No budget data system will alleviate the paranoia exercised and induced where the dominant concept is that the *managers* are to be controlled. (True, a manager can cause problems rather than solve them, but that is another subject.)

A specific feature of managerial performance evaluation must be mentioned here, however parenthetically. There is great virtue to judging managerial skill *in part* by the budget he makes. The figure of merit is a valuable tool for this purpose. But managing the work to the plan must be expected. Any tradition of shifting people between jobs just *after* budget time will reward the unscrupulous who budget overstringently, leaving another to explain and suffer demerit for unavoidable overrun.

The current view

But now to another process to which the APL budget data system can contribute. The inputs are the variances in both productive results and the budget impacts. The process is to assess what effects the sources of variance *and* the chosen corrective measures will have upon year-end-results. The output may be called the *current view*, a very vital input to the decision processes of higher levels of authority. Forming a current view each month is aided enormously by provisioning of the manager through APL with original plans and the actual budget impacts to date.

Such provisioning easily may make the current view feasible. Without the current view process the tendency is to shrug off lumps administered for variances (positive or negative) with a vague resolution to “do better”. The current view focusses attention onto *how* to do better and provides estimates of how much better, just what a manager’s manager needs to do his own best.

Commitment control

Variances, monthly or year-to-date, by line item or rolled up by management unit, are key inputs to many management decision processes. The current view process is a very powerful addition. And one more, very contentious but widely exploited, is *commitment control*. The passive scheme is simple and its APL implementation trivial. One assesses a *budget reserve* as the year-to-date difference between budget-as-planned and expenditures. If, for a line item (or possibly for a cost centre as a whole), there is inadequate reserve, then purchasing authority is suspended, whatever other fiscal integrity controls may have been hurdled. The ultimate candidate for paranoia is the manager who “must get results” but is foreclosed by such a control from buying or retaining the materials or tools vital to those results.

The organization which executes tough-minded examinations of results and costs monthly has scarce need for the commitment control tool. Where intelligent active budget-tracking is not enforced, it substitutes a *budget-control-by-crisis* approach. At best it can left-handedly trigger higher managerial attention to loose practices lower in the ranks. At worst it propagates high level paranoia throughout an organization.

Implementing commitment control

Shucks! It is so easy to mechanize via APL. Your management insists. And you are going to plan and execute so well that the dead man’s throttle of commitment control will be there merely as a backstop. So here are the piece parts to the process.

The *Official Budget* is to be captured, as accommodated to edict, as an item-by-month data array with an itemwise matching descriptive matrix, for each account within each lowest level management unit. A *directory* ties these together to guide rollups to middle and top level summaries.

The *Actuals* start with the planned descriptors and the same data array, to be filled in by month as time passes. The data and new descriptors necessitated by events will be adjusted to actual results to date and reassessed for future months by the current view process.

The *commitment arrays* are like unto the Official Budgets at the outset, except limited to those items which are subject to a standing commitment. Mortgages, rents, leases, maintenance agreements, or outstanding purchase orders exemplify. These arrays are to be adjusted as events conspire (like when those rental increase announcements are made).

Conventional tracking for active control works between the Budget and the Actuals, with monthly and year-to-date outputs. The Current View covers the full twelve months of data, partly actuals and partly the adjusted projections.

Commitment Control derives from Actuals and Commitment Budget arrays, evaluated year to date *and beyond* only. The latter proviso is to ensure against undertaking a commitment of recurring nature which will break the bank downstream. The authority to purchase can then be made a “go-no-go” proposition based on the computed *Budget Reserve*, the difference for the line-item between Official Budget and Commitments.

A single APL file can be built easily to hold all directories of account structure, organization structure, and the descriptors and data for all three types of Budget. The consolidation processes and reporting functions follow easily, formatted to suit taste and organizational features associated with the requisitioning and/or commitment control processes governing the issuance of purchase orders.

Analysis

Each month, we have argued, the APL-processed data on Actuals (budget impacts and productive results) will have been assessed for significance in immediate decision-making processes. Longer and wider views may be formed, helping to improve both short and long term performance, by a critical review of the past year’s performance. The monthly analyses already will have helped through the action-focussing current view process. Now these shorter ranged perceptions may be brought into perspective by reviewing the longer cycle. What early judgements proved faulty as to cause of variance? What control efforts failed to control? Why? What worked and will continue to work through next year?

Clearly we have returned to the cyclical starting line, reentering the planning phase. The condition now should be one of increased awareness of, and astuteness for, the business processes you have been managing. The data, in review, will warn you of *full-year effects*, the seemingly out-of-proportion impact of major midyear changes in goals, plant or processes upon the new year’s budget.

Top down or bottom up?

Line management has been shown to be a continual process of review and decision. At the level of managing resources directly the processes under control feature machinery, raw material and nonmanagerial (that is, non-budget-accountable) staff. This view is totally distinct from that seen from anywhere higher in the organization. It is dealing directly with realities, with all the rocks and hard places. Much of the practical value of APL-based budget planning and tracking aids is literally the enabling of the first-line manager to be truly aware of what he is managing, what the results have been, and what he may do about them. Without this knowledge, decisions may be truly misguided by the “obvious” responses to budgetary or performance pressures from above.

Clearly the well-informed first-line manager is a powerful company decision-maker. Ultimately the corporate results reflect the cumulative result of detailed first-line management decisions. The budget system and its interface to the realities of expenditure, the accounting and payroll systems, need to be first rate at risk of losing what only first-line managers can achieve, direct cost control.

All other viewpoints are top down, covering some subsection of the organization. Each higher viewpoint holds several lower ones in nested overview. The Budgets, Actuals, and Current Views are summaries of data reflecting the management efforts of several or many other persons. The APL consolidations provide much information about consolidated variances. As outlined in the Anthony model of the information system, these summaries do not provide clues for instantaneous action. There are inertial time lags and smoothing effects across units which preclude detailed decision making, favouring instead increasingly general decision making. Strategy and long term plans are the province of the seniormost executives. The security features of APL can and should be invoked as much to preclude downward peeking as lateral or upward. Need-to-know should be implemented between adjacent management levels only in routine upward summaries, to overcome a too human tendency to scrutinize the work of subordinates' subordinates.

Summary

Love for budgets is professed by few. Clinical paranoia is probably as rare. But there are many managers who display symptoms unhappily representative of that state of mind.

Good budget management systems can be built, based on sound comprehensions of what is comprehensive budgeting. Sound managers, properly trained and equipped with information, can and will perform powerfully, at any organizational level. The tools needed to support this power can be built efficiently using APL.

The well trained and equipped manager may never learn to love budgeting. But most will revel in the power that well conceived, well supported, comprehensive budgeting provides over the decision making which affects the outcome of the enterprise.

A CASE STUDY: APL AS THE 40% SOLUTION

Michael J. Waller
Vice-President and General Manager
Computer Consoles Leasing Corporation
Rochester, New York

Computer Consoles Leasing Corporation (CCLC) was founded in March of 1977 as the finance subsidiary of Computer Consoles, Inc. Its principal mission was to finance turnkey systems (e.g. Directory Assistance) to AT&T telephone operating companies. By the end of 1979, it had grown to a level where it held long-term lease receivables of \$39 million.

Up to this time, the administrative process could largely be described as "semi-automated". Information required for bank collateral reporting and accounting was held as global variables in separate workspaces. Due to the volumes and reporting time pressures, some dissimilarities began to emerge between the two types of data. Although they were immaterial for financial reporting purposes, the trend was clearly unacceptable in view of the growth. Finally, mechanisms in place at this juncture met only the minimum requirements of recordkeeping, and denied the Company any ability to either forecast future portfolio characteristics and credit needs, or to exhibit the quality and viability of the existing portfolio to various sources of credit.

Defining the objectives of CCLAS

CCLAS (Computer Consoles Lease Accounting System) was designed to meet very specific and ambitious objectives:

- Employ a common data base which captures all bank and accounting information for each lease event in unique components.
- Produce an input/editing routine that forces rational and consistent judgments and assumptions in booking leases.
- Produce all accounting entries and associated detail for the portfolio on a monthly basis.
- Produce all bank reports on a monthly basis.
- Design the system so that it could easily accommodate future changes in accounting practices or financing rules.

- Design the system so that it could satisfy forecasting needs in the near-term and serve as a basic module for a future comprehensive Corporate Planning System.
- Develop a free report writer that would allow casual access to the data base in order to isolate generic portfolio characteristics and other selective classes of information.

Macro overview of CCLAS

The CCLAS data base contains information pertaining to the financial obligation held by CCLC; e.g., the name and address of the lessee, the lease commencement date, the monthly lease amount, the term of the lease, etc. In all, there may be as many as 71 types of information captured for each lease. Further, the lease components will only vary in size due to the term or length of the lease. Hence, the time factor causes many fields to telescope.

Major software modules that operate on the CCLAS data base are:

- File Maintenance Module:
 - Allows the creation of new lease records based upon input lease characteristics using a smart “expert” or “novice” dialogue.
 - “Deletes” the cancelled portion of any lease record.
 - Removes expired leases and places them in another “dead” file.
- Reporting Module:
 - Produces a summary and detailed listing of current month values of financed leases for the bank compliance report.
 - Generates journal entries and detailed support for all current month portfolio-related items reported in the monthly financial statements.
- Management and Planning Module:
 - Provides casual access to portfolio constitution/characteristics at any point in time or over any period of time on a detailed or summary basis.
 - Offers the means to add forecasts to the existing portfolio in order to project future portfolio characteristics and credit needs.

Micro examination of CCLAS

CCLAS contains approximately 115 programs and subroutines amounting to some 3000 lines of code. After an informal conceptualization period of roughly 2 manmonths, the system was coded, tested, and the existing data base loaded in approximately 5 manmonths.

As noted earlier, some 71 unique types of data were identified for maintenance in the data base. Within these data types, it was further noted that some of these remained the same throughout the life of the lease (static), or changed monthly (dynamic). While devising a means of treating the static data was straightforward, handling of the dynamic data was a considerably more serious undertaking. The available options were to generate dynamic values on a monthly basis using equations and discrete variables, as in the existing “system”, or to generate all of the values at the moment that the lease was entered.

Based upon our limited experience with the process and our attempts to anticipate what the future would hold, the latter approach was chosen. It has proven to be the proper and only workable alternative:

- The “storage approach” proved to be the only means of accommodating the sometimes arbitrary assumptions or actions that were manifested in the initial lease base. In fact, the monthly calculation approach would have been defeated at the outset by this circumstance alone.
- Changes in accounting and financing rules may prove to be sufficiently radical that the calculation approach would prove to be hopelessly complicated and unworkable.
- Because of the projected (and realized) growth in the lease base, the storage approach has resulted in considerably less expense incurred in running the system.
- This approach also provides the flexibility for dividing single lease events for occasional instances when differing valuation techniques must be employed for accounting and bank purposes without losing the integrity of the individual event. That is, $A = B + C$, where A is the source lease event, but only its component parts, B and C are retained.
- As in the prior point, matrix algebra can also be used to produce lease records for uneven payment streams and varying implicit interest rates.

In summary, this design permits the user to create a lease record exhibiting virtually any characteristics that will be precisely reported in the future.

Final comments

CCLAS, although composed in haste, has successfully endured a period of considerable growth at Computer Consoles. At the time that it was initially conceived, the gross value of the portfolio was \$39 million and there were 180 lease records. The value of the portfolio has since grown at a 40% annualized monthly compound rate to \$104 million, and there are now over 1050 records. Required maintenance and enhancements over that 2 1/2-year period have approximated just 6 manweeks.

I would be remiss for not noting also that CCLAS is a general purpose utility, largely independent of any one application. Hence, the real beauty of this design is that given sufficient provocation or incentive, the structure could be reapplied for anything from a personnel data base to a real estate management system. Hence, CCLAS also serves as a tool for satisfying administrative demands in the future.

Finally, I would like to give recognition to my co-conspirator in this development effort, Walter J. (Chuck) McBride, whose intelligence and steadfast determination proved to be the difference.

APL AS A TOOL IN CHEMICAL PRODUCTION

Henning Lervad Andersen
Civil Engineer
Novo Industries Incorporated
Copenhagen, Denmark

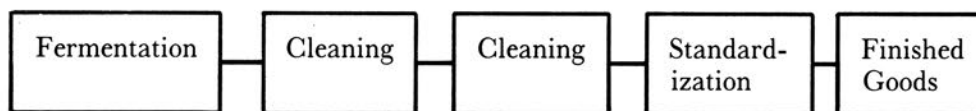
1. About the company

Novo is a company that produces pharmaceuticals and enzymes for industrial purposes. In 1981 sales amounted to about U.S. \$300 million equally shared between pharmaceuticals and enzymes. The company is Danish controlled with its headquarters in Copenhagen, and subsidiaries and information offices in 22 countries. The company employs about 4000 people of whom about 800 are in the subsidiaries and information offices.

2. Enzyme production

The title is "APL as a Tool in Chemical Production". Production is split between pharmaceuticals and enzymes. The APL use I am to cover is only related to the production of enzymes.

Batch production is the most common way of producing enzymes.



Every production process begins with fermentation resulting in a well-defined limited batch of enzymes. Then the batch goes through a purification process where the impurities are removed. The next step is standardization where the batch by evaporation or excipient becomes a well-defined finished article. Finally the enzymes are packed.

The composition of steps and the amount of steps in the production process varies in accordance with what enzyme is to be produced, just as batches of the same enzyme may go through various steps depending on the analysis of the batch and the present capacity situation.

The fermentation process starts with a well-defined batch, but later on a split up of the batch may be necessary depending on the capacity of the individual step, or a mix

with other batches takes place in order to obtain a quantity that corresponds to the capacity of the step.

3. APL systems in Enzyme Production

In Enzyme Production a number of APL systems are used of which the following will be described:

- 1) Production Control
- 2) Quality Control of Finished Goods
- 3) Production Planning

At present these three systems are independent systems. System 1 and 2 are used by Enzyme Production in their daily work on a number of APL-terminals (12), placed in each Production Department and in the Production Approval Department. System 3 (production planning) is used quarterly by the Planning Department.

4. Production control

This system which is far the most comprehensive of the three systems has been used by the Production Department for 18 months. The aim of this system is to follow each enzyme batch from the beginning of the fermentation process till it leaves Novo. Thus it is possible to reconstruct the production process for a special batch of finished enzymes, and then determine what batches of semi-finished products were used at each step. Likewise it is possible to reconstruct, for instance, a fermentation batch, and determine which batches of finished products contain parts of this batch. Furthermore the system can print out stock in process and finished stock whenever it is needed, just as traditional periodic production reports can be prepared. In the long run, i.e., when the data material has reached a sufficient size, the system can be used for analyzing the connections in the production process, e.g., connections between the value of certain parameters in the production process and the value of parameters in the finished goods which determine the quality of the products.

4.1 The function of the system

The unit in a system is one process step for a product. With about 20 different enzyme products and 5-15 process steps per product a quantity of 200-300 various process steps is reached. Each Production Department sets up the process steps that can be illustrated with the following example:

Figure 1
Step Survey

```
PRODUCT : 399 CEREFLO
STEP    : 45  EXAMPLE

DELIVER TO : 670 680 4548

RAW MATERIALS
-----
1  KALCIUM CHLORID      1010603      2.10/KG
2  NALCO                1010158      12.00/KG

UNIT OPERATIONS
-----
3  DESLUSGE HI          START
4  DESLUDGE HI          END
5  STEAM
6  WATER

PROCESS PARAMETERS
-----
7  TEMPERATURE (C)
8  TIP VELOCITY

ANALYSES
-----
9  WEIGHT (TON)
10 DENSITY (G/ML)
11 ACTIVITY (KNE/G)
```

A step can be characterized by:

- a) product
- b) step number and name
- c) the materials used for the step in question
- d) the unit operations that form part of the step
- e) the process parameters entered per batch
- f) identification of the next step in the production process.

Figure 1 shows a printout of a step using 2 kinds of raw materials, 1 unit operation, 1 process parameter and 3 analyses. All parameters, for which data are entered, have been identified by a number that is unique for the step.

When a batch of enzymes is produced, the data for the batch is available at various points of time, so more than one person in the department can be responsible for entering the various data for the batch. Further to this it is necessary that typing in data for the same batch—which is done by various people—can be done independently. Therefore we operate with data sheets that can be defined as:

Figure 2

<i>PRODUCT</i>	: 399
<i>STEP</i>	: 45
<i>DATA SHEET NO.</i>	: 5
<i>DATA NO.</i>	: 1 4 6 7 10

and looks like the following:

Figure 3

<i>PRODUCT</i>	: 399	<i>CEREFLO</i>
<i>STEP</i>	: 45	<i>EXAMPLE</i>
<i>DATA SHEET NO.</i>	: 5	

	BATCH DATE					
	BATCH NO.					
1	KALCIUM CHLORID					
4	DESLUDGE HI END					
6	WATER					
7	TEMPERATURE (C)					
10	DENSITY (G/ML)					

Data for each batch are written in pencil on the data sheet in a separate column. The data of a process step can be distributed on an arbitrary number of data sheets. When inserting the data from a data sheet you just have to state:

- product
- step
- data sheet number

and then you have to insert the data by column, but *don't* have to identify the various figures as the system knows what data and in which order they appear on each data sheet, (see below):

Figure 4

<i>PRODUCT</i>	:	399
<i>STEP</i>	:	45
<i>DATA SHEET NO.</i>	:	5
<i>BATCH DATE</i>	:	811106
<i>BATCH NO.</i>	:	ADB714
1 <i>KALCIUM CHLORID</i>	:	14.5
4 <i>DESLUDGE HI</i> <i>END</i>	:	1106.1430
6 <i>WATER</i>	:	715
7 <i>TEMPERATURE (C)</i>	:	38.5
10 <i>DENSITY (G/ML)</i>	:	1.124

In the system there are a number of reports for:

- stock
- cost analyses
- step performance
- unit operation performance

Furthermore a report generator has been put into the system by which the user himself can define what data he will put together in columns, the breadth of the column, the headlines of the columns, etc. This report generator together with plots and statistical tools forms the part of the system which is to contribute to the analysis of the data across the steps of a product in order to throw light on the relations between parameters. These relations are difficult to calculate manually due to the mingling and splitting up of batches.

5. Quality control

All finished batches have to be approved by the Product Approval Department. Each product (about 100) has a product specification indicating what analyses (about 30) have to be made of the finished goods and within what limits the results are to be.

The aim of the APL-system for quality control is:

- a tool for maintaining product specifications
- to check that all analyses have been made
- to check that the results of the analyses are within the specification limits and to print out certificates for the approved batches
- to collect data for a longer period

The system contains statistical aids that are to analyse whether there are any trends in the results of the analyses of a product. If a batch does not comply with the product specification, the system will tell you if there are other products for which the batch fulfills the product specification.

The system is for the moment used by our enzyme factories in the United States, Switzerland and Denmark, all running on our computer through the I.P. Sharp Network. Therefore it is natural that the system is used for consolidation of data into production reports.

6. Production planning

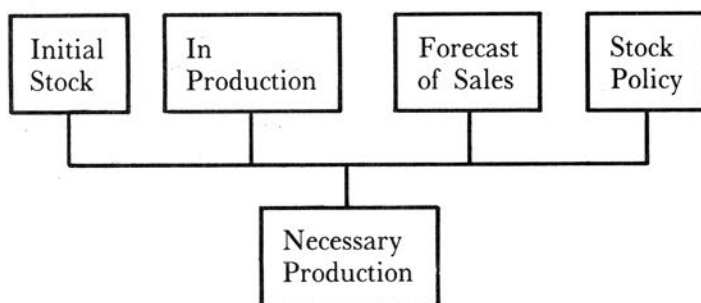
The system for production planning is used once every quarter by the central department for Production Planning. By means of this system the production plan for a planning period of four quarters is determined.

The production plan indicates quarterly:

- the quantity to be produced of each product
- where the production is to take place

as the same capacities—placed at various plants—can be used for the production of different enzyme products.

6.1 How the system works



The first step in the preparation of the production plan is to calculate the quantity that is to be produced in each quarter of the planning period. The basis for this calculation is the registration of:

- 1) the present stock of each product
- 2) present production that will be finished within the planning period
- 3) the stock policy which states the amount of each product being stocked at the end of every quarter
- 4) a forecast of the sales of each product per quarter

Based on the above it is calculated what is to be produced per quarter in order to fulfill the expected demand and to observe the stock policy.

Then it is to be decided where the production can be done with the present capacity. As an aid in this decision process the following registered information is used, as it is primarily the fermentation capacity which has to be optimized:

- 1) each product's use of capacity, i.e., the length of the fermentation period and how many enzyme units each fermentation produces
- 2) in which fermentation tank the products can be produced. Every fermentation tank can produce several products, however, there are a few limits in the flexibility due to various reasons
- 3) the fermentation capacity at each location
- 4) product priority—the importance of fulfilling the stock policy differs from product to product, and therefore the products are ordered according to this

Based on this information the system creates a production plan following a heuristic method. The aim of this plan is to produce as much as possible within the restrictions given.

After this plan some of the products will not be fully produced because of lack of capacity, but at the same time some of the fermentation tanks will not be fully used because of the restrictions stated. Next step in making the production plan is made in communication between the system and the user. For the products that could not be fully produced you get the following printout:

Figure 5

*HIGHEST PRIORITY PRODUCT NOT FULLY PRODUCED : 3569985 **ALCALASE 1.5 M*

<i>POLICY</i>	:	2650.9	2761.4	3110.2	2569.3
<i>RESULTING</i>	:	1839.1	1949.6	2298.4	1757.5
<i>NESC. PRODUCTION</i>	:	2762.7	1924.3	2058.2	1217.0
<i>PRODUCTION</i>	:	1950.9	1924.3	2058.2	1217.0
<i>SHORTAGE</i>	:	811.8	0.0	0.0	0.0

IS THIS SHORTAGE OK ? N

OPTION : ?

OPTIONS ARE:

LOWER STOCK POLICY

LOWER ROLLING OFF PLAN

DELAY TO FREE CAPACITY

DELAY

MOVE AHEAD TO FREE CAPACITY

MOVE AHEAD

USE FREE CAPACITY

As can be seen we did not manage to produce 811.8 tons in the first quarter of the planning period and that is why the stock policy is not fully met in any of the quarters. Finally, the printout gives us a number of possibilities for modifying the plan manually.

By preparing the production plan according to this system you get a rough quarterly plan. The aim of this plan is primarily to decide if there is enough capacity to meet the demand and the stock policy. The daily and weekly production plan is prepared in detail by each Production Department.

Why APL?

APL is used to solve the above mentioned tasks which for at least one of the systems contains a very large amount of data, and for two of the systems a large number of transactions are included.

APL is used for the following reasons:

- The systems are not and will probably never reach a steady state. There is still a demand for new facilities and modification in the existing systems.
- The development of the systems has been a “trial and error” process. The users could not sign a system specification, but had to see something working before they could decide what they wanted.
- The alternative language was COBOL. With the existing backlog for the COBOL systems and considering the productivity of COBOL I guess the systems would never have been developed.
- The users had other APL systems and were satisfied with them.
- The factories (plants) are situated in several countries; however, with the network of I.P. Sharp it is possible to use the same systems on the same machine.
- APL is a language in which the EDP department has expertise.

The future

The use of EDP for production control has been started up. Today at Novo we use APL widely for registration and further treatment of production data, and minicomputers for controlling and regulation purposes at the individual steps of the process.

The next steps in the development within this area will be to integrate the APL systems so that, e.g., Production Planning can take advantage of the realized production figures when deciding each product's use of the unit operations instead of using the registered theoretical figures. Further to this, the communication between minicomputers and APL will be developed, i.e., a direct data collection for APL of the relevant production data that have been registered by the minicomputers.

LET THE NUMBERS DO THE TALKING—WITH MAGIC

David Keith
Manager, Data Base Services
I.P. Sharp Associates Limited
Toronto, Ontario

Introduction

APL is a superb language for the analysis of numeric data. It is rich in primitive functions, it processes arrays of data as easily as it does single elements, and it is not a difficult language to learn. How then can there be room in an APL environment for a language like MAGIC, which uses simple English words in a way which appears to shelter the user from the benefits which APL can offer? The truth is that MAGIC does *not* shelter the user from the benefits of APL—its success is most definitely related to its dependence on the APL environment.

Although APL may be powerful and easy to learn, it is still a full-blown computer language, and there are a lot of people who will never learn a computer language, be it BASIC, COBOL, APL, or whatever. But many of those people do have a legitimate requirement for the analytical capability of computing—especially in the business environment, where several stringent conditions often preclude the use of computers. These conditions include such items as:

- Very little time is available for complex analysis. Results are needed quickly.
- No one is available who understands the issue at hand *and* is a computer programmer.
- The data required to provide in-depth analysis may not be available.
- Incomplete understanding (i.e., weak presentation) of pertinent data can lead to a decision which is financially disastrous.

This paper shows how MAGIC, in an APL environment, provides an effective management tool for the analysis and presentation of business-oriented data (private or public) by non-programmers. In doing so, it relies primarily on data from the over eighty public data bases available on the I.P. Sharp system. Although not discussed in this paper, you can use MAGIC to build your own private data bases, so that the techniques described here can be applied equally effectively to your own data, or to a combination of your own data and public data.

The I.P. Sharp Public Data Bases

The list of data bases and the volume of data available to users of the I.P. Sharp timesharing system continues to grow at a prolific rate. There are over 30 million time series scattered amongst the following data bases:

Financial Data Bases

Australia

- Australian Bank Bill Rates
- Australian Stock Exchange Indices
- Australian Commodities
- Australian Financial Data Base
- Australian Funds Markets
- Commonwealth Bank Bond Index
- Sydney Stock Exchange STATEX Service
- Sydney Stock Exchange Share Prices
- Currency Exchange Rates (Melbourne)

Canada

- Agricultural Commodities (includes U.S. data)
- Canadian Bonds
- Canadian Stock Options
- Canadian Department of Insurance
- Financial Post Corporate Data
- Financial Post Securities Data
- North American Stock Market
- Toronto Stock Exchange 300 Index
- Chartered Banks Monthly Statement of Assets and Liabilities
- Chartered Banks Quarterly Income Statement
- Bank of Canada Weekly Financial Statistics
- Money Market Rates
- Currency Exchange Rates (Toronto)

United States

- Commodities
- U.S. Stock Options
- North American Stock Market (NYSE, ASE)
- Federal Reserve Board Weekly Statistics
- Money Market Rates
- Currency Exchange Rates (New York)

Other

- Financial Times Actuaries Share Indices
- Financial Times Share Information
- U.K. Commodities
- U.K. Fund Monitor Unit Trusts and Insurance Bonds
- Eurocurrency Money Market Rates
- Currency Exchange Rates (London, Paris, Copenhagen, Helsinki, Vienna)

Economic Data Bases

Australian Bureau of Statistics
Australian Economic Statistics
Australian Sector Cash Flow
Australian Input-Output Data Base
Statistics Canada's CANSIM Data Base
CITIBASE—U.S. Economic Data
National Planning Association U.S. Economic Data
National Planning Association U.S. Demographic Data
United Kingdom Central Statistical Office
Bundesbank Economic Data for Germany
S.J. Rundt World Risk Analysis Package
Business International Historical Data
Business International Country Forecasts
OECD Main Economic Indicators
OECD Quarterly and Annual National Accounts
IMF International Financial Statistics
World Bank Debt Tables

Aviation Data Bases

Civil Aeronautics Board

- Form 41
- ER586 Service Segment
- Origin-Destination
- Commuter Origin-Destination
- Combined T9/Service Segment

Official Airline Guide Flight Schedules
Association of European Airlines
Canadian Operating Statistics
ICAO Traffic Statistics
AISL Aircraft Accidents

Energy Data Bases

Weekly Statistical Bulletin (API)
Heavy Fuel Oils (DOE)
U.S. Petroleum Imports (API)
International Petroleum Annual (DOE)
Liquefied Petroleum Annual (DOE)
Monthly Energy Review (DOE)
Monthly Petroleum Statistics Report (DOE)
Monthly Supply and Disposition Report (DOE)
State Energy Data System (DOE)
United Kingdom Energy Trends
Hughes Oil Rig Count
Petroleum Argus Daily Market Report
Petroleum Argus Prices
Petroleum Intelligence Weekly

- Production
- Crude Oil Prices
- Spot Product Prices

Petroleum News Data Service

A characteristic of almost all of the data bases is that they are composed of time series. A time series is a set of observations that is evenly spaced over time. Examples of time series are the daily closing price of gold on the London Bullion market, or the number of passengers carried by month on DC-10s between Chicago and Miami. By definition, a time series must have a start date (e.g., August 1, 1978), an end date (e.g., yesterday), and a frequency (e.g., daily). The number of observations in a time series may vary from a few, to thousands. Clearly, many of the events that occur in this world may be recorded as time series. But whether it is the national accounting system for a country (e.g., GNP for Germany), or the private accounting system for your company (e.g., foreign exchange gain or loss), the important point is that the same analysis techniques can be effectively applied in either case. It is also important that users of data in the public data bases have a consistent method of data access. That is to say, the same workspace, and the same style of access, should be applicable no matter what the data base. Data from several data bases must be able to be mixed together in the same application.

MAGIC is a language which provides a simple yet powerful mechanism for tying all of the data mentioned above together. It is not a system. It is a set of modules (APL functions) which reside in an APL workspace. Some of the functions allow you to set a timeframe. Others allow you to access data from the data bases, while still others permit the manipulation of that data, and the formatting of the results in tabular or graphical form. Because of its modularity, new capabilities or new data bases can be added to MAGIC without requiring changes to the existing software. More advanced users can define their own APL functions consisting of MAGIC statements. These can be saved and re-executed as required, for example, to incorporate data from new data bases into an existing application. Given these rather bold statements about MAGIC, it is surprising that MAGIC can accomplish all of this using simple English-like commands.

A Little MAGIC

As an analyst, you may be interested in following the price movement of certain stocks on the New York Stock Exchange. Like APL, it usually takes more English statements to describe what is happening in MAGIC commands than the commands themselves. Here are some typical MAGIC statements to produce a basic listing of data for Exxon Corporation on the New York Stock Exchange.

```
)LOAD 39 MAGIC
SAVED 22.14.21 08/20/82

1 2 3 4 5 DAILY,DATED 1 7 82 TO 31 7 82
AUTOTITLE
AUTOLABEL
'H' TABLE 'NXON' NASTOCKD 4 5 6 9
```

	EXXON CORPORATION			
	HIGH	LOW	CLOSE	VOLUME
1JUL82	27.75	27.25	27.38	430,700
2JUL82	27.25	27.00	27.13	502,400
5JUL82	27.25	27.00	27.13	
6JUL82	27.13	26.75	26.88	420,700
7JUL82	27.00	26.63	27.00	357,900
8JUL82	26.88	26.25	26.63	672,200
9JUL82	27.00	26.63	26.88	631,000
12JUL82	26.88	26.25	26.38	898,000
13JUL82	26.63	26.38	26.50	916,000
14JUL82	26.50	26.13	26.50	957,100
15JUL82	26.75	26.00	26.63	600,500
16JUL82	26.88	26.50	26.88	405,600
19JUL82	27.00	26.63	26.63	433,400
20JUL82	26.88	26.50	26.63	458,300
21JUL82	26.88	26.50	26.50	441,200
22JUL82	27.00	26.50	27.00	524,500
23JUL82	27.00	26.38	26.63	1,082,800
26JUL82	26.75	26.50	26.75	368,100
27JUL82	26.75	26.50	26.63	544,900
28JUL82	26.50	26.13	26.25	417,000
29JUL82	26.38	26.00	26.25	453,700
30JUL82	26.25	25.88	26.00	623,500

In the above example, after 39 *MAGIC* is loaded, four *MAGIC* statements are typed into the system. The first sets a timeframe for all business days of July, 1982. The second and third lines request automatic titles and labels to be retrieved with any data base access. And the fourth contains a single statement to retrieve the data of interest (using access function *NASTOCKD*), and to output the results (using *TABLE*). The 'H' to the left of *TABLE* requests that highlight bars be used to separate the labels from the numbers. It is clear from the above example that *MAGIC* provides a trivial method to list any of the data in the public data bases.

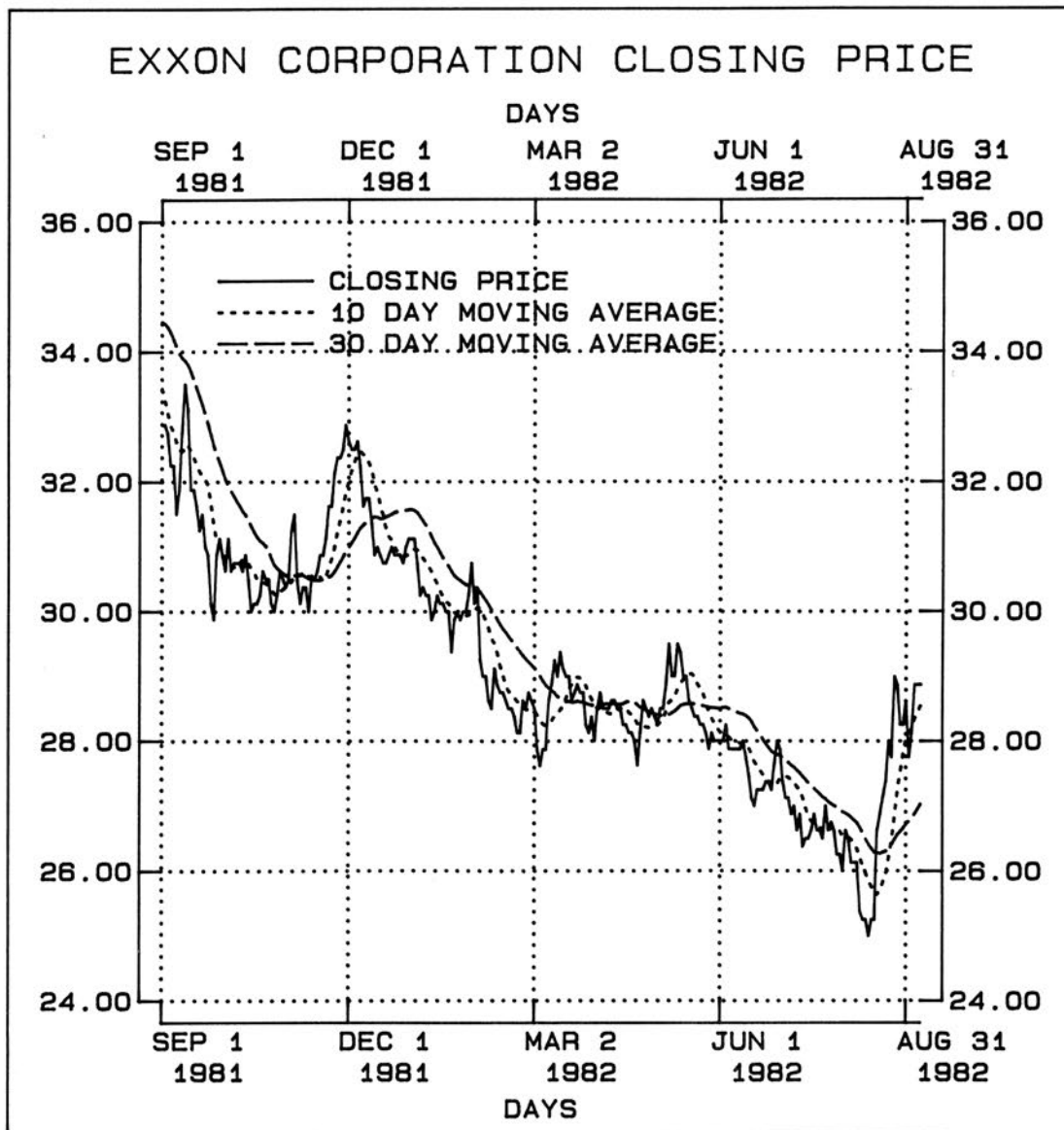
The next example shows *MAGIC* being used in a slightly more sophisticated way—one that requires the use of *MAGIC*'s moving average function on data retrieved from the data base. A very simple method used by technical analysts of the stock market is to watch cross-over points of the 10- and 30-day moving averages of a stock's closing price. If the 10-day average moves above the 30-day average, it is time to buy, and if it moves below the 30-day average, it is time to sell. Using the following *MAGIC* statements, it is possible to investigate how well this theory applies to Exxon Corporation's closing price on the NYSE for the past year:

```

RESETOPTIONS
1 2 3 4 5 DAILY,DATED 1 7 81
PUT 'NXON' NASTOCKD 6
PUT 10 MOVAVG ITEM 1
PUT 30 MOVAVG ITEM 1
LABEL 'CLOSING PRICE,10 DAY MOVING AVERAGE,30 DAY MOVING AVERAGE'
TITLE 'EXXON CORPORATION CLOSING PRICE'
ΔSUPERPLOT 'RESET,ALL/TERM,HP7221A/FONT,TITLE,1.5'
ΔSUPERPLOT 'COLOUR,1 2 2/STYLE,SOLID,SDASH,LDASH'
ΔSUPERPLOT 'GRID,TIKS,1,DOT'
'LRTBH' PLOT ABOVE,ONLY 1 9 81

```

Figure 1



The first seven statements above establish a timeframe, retrieve the data, perform the moving average calculations, and specify the appropriate descriptive text. The next

three statements stipulate the parameters that are to be used in conjunction with SUPERPLOT, MAGIC's business graphics package. The last statement requests the plot. Each letter to the left of *PLOT* requests various SUPERPLOT options. 'LRTBH' specifies that the plot is to have *left, right, top, and bottom* axes, and a *highlight* box around the plot.

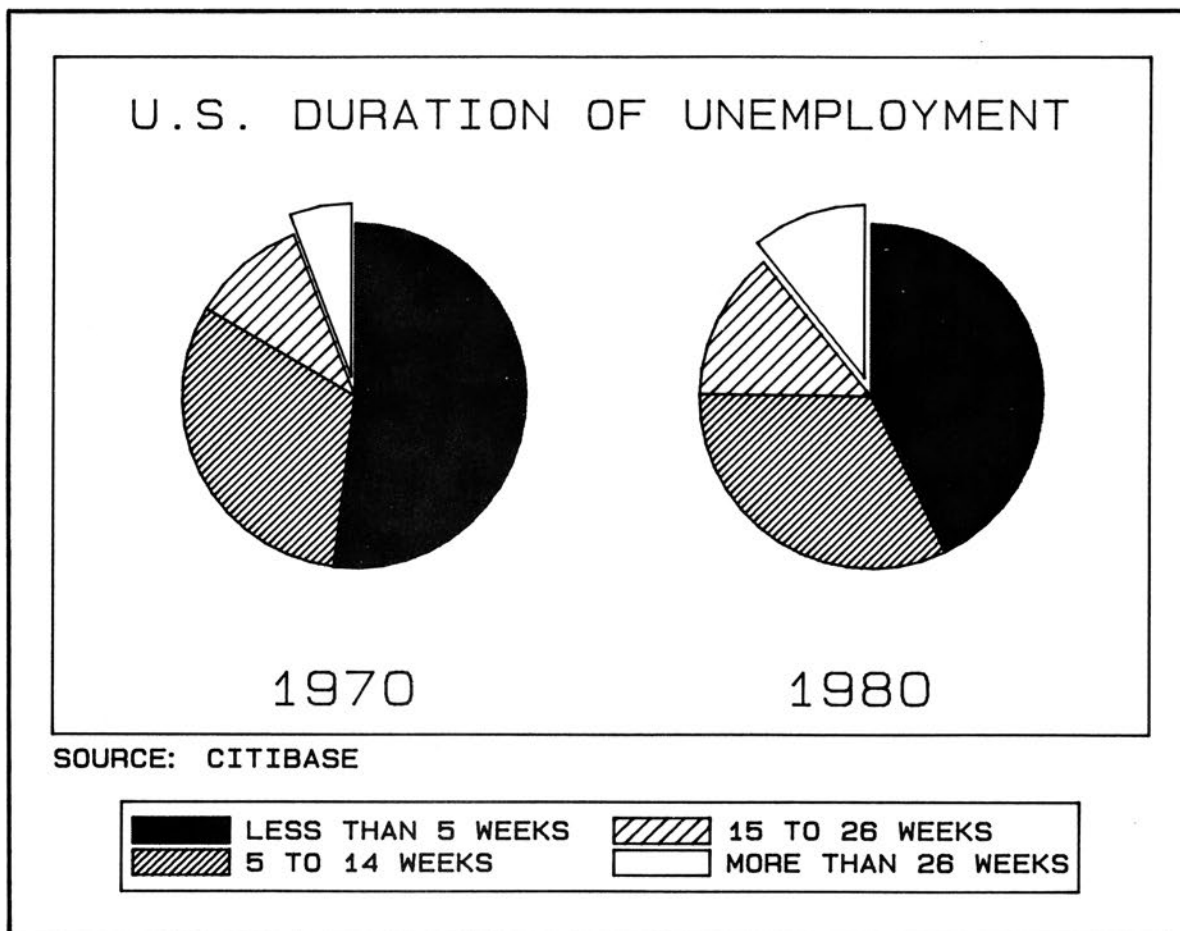
SUPERPLOT can be used equally well to produce pie charts, showing differences over time for a variable you may be trying to highlight, such as the duration of unemployment for more than 26 weeks, using Citibank's CITIBASE data base.

```

RESETOPTIONS
YEARLY,DATED AT 70 80
AVERAGES
TITLE 'U.S. DURATION OF UNEMPLOYMENT'
LABEL 'LESS THAN 5 WEEKS,5 TO 14 WEEKS'
LABEL '15 TO 26 WEEKS,MORE THAN 26 WEEKS'
ΔSUPERPLOT 'RESET,ALL/TERM,HP7221A'
ΔSUPERPLOT 'FONT,TITLE,2,XAXIS,2/XLABEL, '
ΔSUPERPLOT 'TYPE,PCH,PCH,PCH,PCO'
ΔSUPERPLOT 'SHADING,1,1,45,1/2,2,45,4/3,3,45,8'
FOOTNOTE 'SOURCE: CITIBASE'
'H□' PLOT CITIBASE 'LHU5,LHU14,LHU26,LHU27'

```

Figure 2



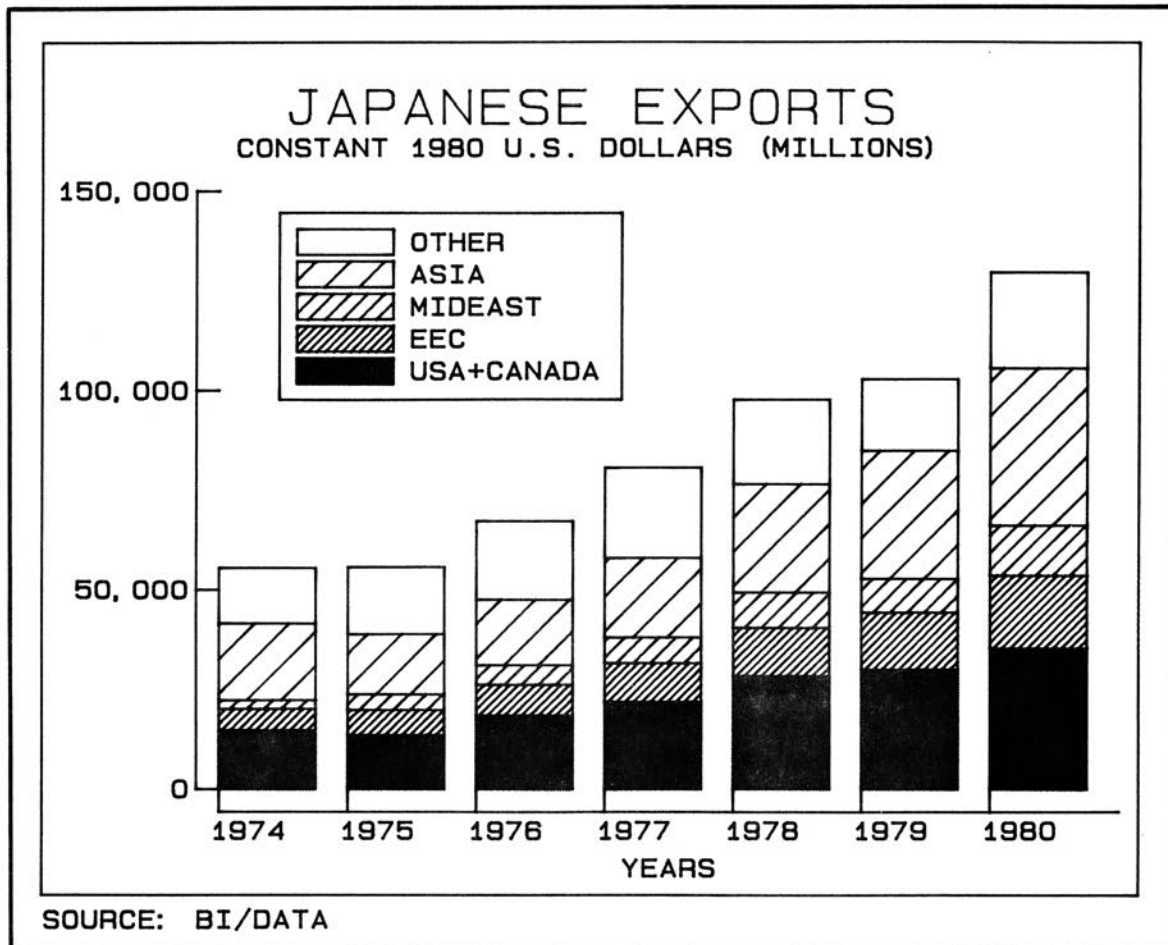
Another effective method of illustrating changes over time is the stacked histogram, as the following plot shows, using data drawn from Business International's BI/DATA data base.

```

RESETOPTIONS
YEARLY,DATED 74 TO 80
SCALE 1000000
PUT SUM 'USA/80/' BIHIST 'USCAN/IMJAP'
PUT SUM 'USA/80/' BIHIST 'EEC/IMJAP'
PUT SUM 'USA/80/' BIHIST 'MIDEAST/IMJAP'
PUT SUM 'USA/80/' BIHIST 'ASIA/IMJAP'
PUT('USA/80/' BIHIST 'JAP/EXPTL') MINUS SUM ABOVE
TITLE 'JAPANESE EXPORTS'
TITLE 'CONSTANT 1980 U.S. DOLLARS (MILLIONS)'
LABEL 'USA+CANADA,EEC,MIDEAST,ASIA,OTHER'
FOOTNOTE 'SOURCE: BI/DATA'
ΔSUPERPLOT 'RESET,ALL/TERM,HP7221A/FONT,TITLE,2 1'
ΔSUPERPLOT 'TYPE,SHI/SHADING,1,BOT,45,1/2,1,45,4/3,2,45,8/4,3,45,16'
'H' PLOT ABOVE

```

Figure 3



Many of the public data bases contain an incredible amount of detail. For example, the T9S data base in the aviation area contains monthly detail about each carrier's activities in each of the thousands of city pairs flown by U.S. airlines. This level of detail permits, for example, a market share calculation for any single city pair as follows:

```

RESETOPTIONS
NOYEAREND ◇ MONTHLY,DATED 8 81 TO 12 81
TITLE 'NON-STOP TRAFFIC BETWEEN NEW YORK AND BOSTON'
CARRIERS+IND,T9SCAR 'NYC-BOS'
PASSENGERS+ 'NYC-BOS' T9S,CARRIERS,33
RANK+VPASSENGERS[;5]
CARRIERS+CARRIERS[RANK]
PASSENGERS+PASSENGERS[RANK;]
LABEL 'REVENUE PASSENGERSα' ◇ LABEL NAME CARRIERS
PUT PASSENGERS
LABEL 'αMARKET SHAREα' ◇ LABEL NAME CARRIERS
PUT 100 TIMES PASSENGERS DIVIDED BY PASSENGERS[1;]
LABELWIDTH 30
'H' DISPLAY ABOVE

```

NON-STOP TRAFFIC BETWEEN NEW YORK AND BOSTON

		AUG/81	SEP/81	OCT/81	NOV/81	DEC/81
<u>REVENUE PASSENGERS</u>						
IND	INDUSTRY TOTAL	261,726	284,828	321,810	304,548	266,979
EA	EASTERN - DOMESTIC	138,789	168,480	196,816	198,692	177,705
PE	PEOPLE EXPRESS	23,129	25,207	27,417	20,732	21,993
PI	PIEDMONT	15,372	16,037	17,517	16,327	16,006
WO	WORLD	13,435	11,930	10,547	9,867	12,766
DL	DELTA - DOMESTIC	11,798	12,854	14,457	12,886	12,691
TW	TRANS WORLD - DOMESTIC	19,747	12,996	12,237	12,278	10,396
AA	AMERICAN - DOMESTIC	5,553	5,188	5,793	5,573	6,251
CO	CONTINENTAL - DOMESTIC			583	3,235	3,672
NW	NORTHWEST - DOMESTIC		2,511	4,359	2,954	3,264
AL	USAIR	3,298	3,140	3,026	2,974	1,774
UA	UNITED - DOMESTIC					249
NY	NEW YORK AIR	29,633	24,458	26,592	19,030	212
NE	AIR NEW ENGLAND	972	2,027	2,466		
QH	AIR FLORIDA					
<u>MARKET SHARE</u>						
IND	INDUSTRY TOTAL	100	100	100	100	100
EA	EASTERN - DOMESTIC	53.03	59.15	61.16	65.24	66.56
PE	PEOPLE EXPRESS	8.84	8.85	8.52	6.81	8.24
PI	PIEDMONT	5.87	5.63	5.44	5.36	6.00
WO	WORLD	5.13	4.19	3.28	3.24	4.78
DL	DELTA - DOMESTIC	4.51	4.51	4.49	4.23	4.75
TW	TRANS WORLD - DOMESTIC	7.54	4.56	3.80	4.03	3.89
AA	AMERICAN - DOMESTIC	2.12	1.82	1.80	1.83	2.34
CO	CONTINENTAL - DOMESTIC			0.18	1.06	1.38
NW	NORTHWEST - DOMESTIC		0.88	1.35	0.97	1.22
AL	USAIR	1.26	1.10	0.94	0.98	0.66
UA	UNITED - DOMESTIC					0.09
NY	NEW YORK AIR	11.32	8.59	8.26	6.25	0.08
NE	AIR NEW ENGLAND	0.37	0.71	0.77		
QH	AIR FLORIDA					

There are several interesting features of MAGIC and APL in the example:

- line 5 makes use of the cross-reference function *T9SCAR*. It automatically returns as its result the carriers that were active for at least one of the periods in the timeframe. This result is stored in the APL variable *CARRIERS* for later use. Cross-reference functions such as this become increasingly important as the size of a data base increases.
- lines 7 and 8 show the use of the APL downgrade function (Ψ) to sort the passenger count data according to the fifth column (data for December, 1981). Although this step is not absolutely necessary, it does illustrate why MAGIC benefits from being in the APL environment.
- line 13 uses the MAGIC words *DIVIDED BY*. In this case, the left side to *DIVIDED BY* is the table of passenger counts for each airline, while the right side is a single vector of data for row 1 of *PASSENGERS* (i.e., the industry total data). In such a case, MAGIC easily handles the problem encountered when a matrix is divided by a vector. It does so by extending the vector as appropriate. *DIVIDED BY* also handles the case of division by zero, returning a zero in the appropriate spot of the result.

Understanding MAGIC

Although an understanding of APL is not required to use MAGIC successfully, an understanding of how MAGIC functions in an APL environment can assist and enhance its use. When you enter MAGIC statements at the keyboard, they must be syntactically correct from the APL point of view. If not, APL will reject them with a normal APL error message. This means that a statement like:

```
DISPLAT(ITEM 1) PLUS(ITEM 2) results in:  
VALUE ERROR  
DISPLAT(ITEM 1) PLUS(ITEM 2)  
^
```

If you are a novice user, this can be somewhat frustrating, since there are no friendly prompts telling you what is wrong and suggesting ways to fix it. But there are many advantages to MAGIC's open-ended nature:

- **simplicity**—very useful results can be achieved by entering very few statements while in immediate APL execution mode. There is no need to learn how to write and edit APL functions. Yet these, and other APL system capabilities are available when you are ready for them.
- **flexibility**—if MAGIC doesn't quite have what you want, you can often include your own APL as required, providing much more scope for complex problems.

- **growth**—new capabilities and new data bases can easily be “tacked on” to MAGIC without requiring changes to the parts of MAGIC that already exist. MAGIC programs written seven years ago are still executable today.
- **cost**—by relying on APL to perform much of the error checking, there is less CPU expended in deciphering and executing MAGIC statements.

There are many state settings (or options) which can affect the performance of MAGIC. Some or all of these can be changed as required. For example, the default setting of *YEAREND* causes any MAGIC data base access function to return values for each period in a timeframe, *as well as* extra values for year-end periods. If you don't want these year-end figures to occur, you can inhibit them by entering *NOYEAREND* before the data access. Once set, an option stays in force until it is reset. There are currently 20 such options in MAGIC, and all of them are reset to default values when you enter *RESETOPTIONS*, or when you load the MAGIC public workspace. The one option which is not preset is the timeframe. Usually, but not necessarily, you set a MAGIC timeframe which has the same frequency as the data you want to access. Examples of MAGIC timeframe settings are:

```
5 WEEKLY,DATED 1 82 TO 6 82
MONTHLY,DATED 1 78 TO 12 81
YEARLY,DATED 1980 TO 2001
```

When a timeframe, such as one of the above, is set, it stays set. All MAGIC data base access functions return a value for each of the periods set. This may require that it return zeroes if there are periods in your timeframe which are not available in the data base, or that the data in the data base is compacted across time to match your timeframe (e.g., when you set a yearly timeframe, but access a quarterly time series).

For example, if you set the following quarterly timeframe and then access the series for Canada's population from Statistics Canada's CANSIM data base, four values are returned.

```
NOYEAREND
QUARTERLY,DATED 1 81 TO 4 81
CANSIM 'D1'
24229600 24295800 24365000 2443310
```

If you change the timeframe to the four quarters of 1982, two zeroes are part of the result, since those figures are not yet available.

```
QUARTERLY,DATED 1 82 TO 4 82
CANSIM 'D1'
24498900 24560600      0      0
```

If you want to look at Canada's population for several years, with a yearly frequency, then you should tell MAGIC what rule you want to use to convert the quarterly values to yearly values (in this case, the last non-zero value for each year).

```

YEARLY,DATED 79 TO 82
LASTVALUE
CANSIM 'D1'
23860700 24162200 24433100 24560600

```

Because MAGIC access functions (such as the above *CANSIM* example) return results, you can assign them to APL variables for further manipulation if so desired. But what is more important from the MAGIC point of view is that you can pass those results on to other MAGIC functions. Thus, a statement such as:

```
DISPLAY 4 MOVAVG CANSIM 'D1'
```

is an APL statement which performs the very useful function of retrieving the time series for Canada's population, calculating a four-point moving average on that, and then displaying this result on the terminal.

More experienced users of MAGIC can combine several MAGIC statements into APL functions (or programs). An advantage of doing this is that the same program can be used over and over with different parameters, without having to re-enter each MAGIC statement each time you want to run the program. A second advantage is that you can save the program for future use, either by use of the MAGIC Storage System (MSS), or by saving the program as part of an APL workspace.

For example, you could write the following program which would allow you to retrieve and print any of the data in I.P. Sharp's Money Market Rates data base, for any timeframe, as follows:

```

▽ LISTRATES
[1]  RESETOPTIONS
[2]  AUTOLABEL
[3]  AVERAGES
[4]  'ENTER TIMEFRAME (MAGIC FORMAT)'
[5]  □
[6]  'ENTER MONEY MARKET RATE CODE(S)'
[7]  CODES←□
[8]  TITLE 'MONEY MARKET RATES'
[9]  COLWIDTH 11
[10] 'ALIGN PAPER'
[11] 'H' TABLE MRATE CODES
▽

```

You could use this program to retrieve and output data:

```

LISTRATES
ENTER TIMEFRAME (MAGIC FORMAT)
□:
  4 WEEKLY,DATED 1 6 82 TO 19 8 82
ENTER MONEY MARKET RATE CODE(S)
UPRIH,UPRIL,UTBLC30,UTBLC60,UTBLC90
ALIGN PAPER

```

MONEY MARKET RATES

	U.S. PRIME-HIGH	U.S. PRIME-LOW	U.S. T-BILL - 30 DAYS	U.S. T-BILL - 60 DAYS	U.S. T-BILL - 90 DAYS
3JUN82	16.50	16.33	12.29	12.31	12.65
10JUN82	16.50	16.00	11.98	12.10	15.52
17JUN82	16.50	16.40	12.19	12.73	13.13
24JUN82	16.50	16.50	11.93	12.73	13.64
1JUL82	16.50	16.50	12.50	13.05	13.72
8JUL82	16.50	16.50	11.98	12.36	12.37
15JUL82	16.50	16.50	11.41	11.99	12.20
22JUL82	16.20	16.20	10.84	11.50	10.70
29JUL82	16.00	15.70	9.76	10.21	10.95
5AUG82	15.25	15.13	8.89	9.58	10.25
12AUG82	15.00	15.00	9.19	9.58	9.76
19AUG82	15.00	14.25	7.75	8.00	7.72

You could run the same program for a monthly timeframe as well. In that case, daily and weekly values are converted to average monthly values due to line [3] of *LISTRATES*.

LISTRATES
 ENTER TIMEFRAME (MAGIC FORMAT)
 □:
 MONTHLY,DATED 1 81 TO 12 81
 ENTER MONEY MARKET RATE CODE(S)
 UPRIH,UPRIL,UTBLC30,UTBLC60,UTBLC90
 ALIGN PAPER

MONEY MARKET RATES

	U.S. PRIME-HIGH	U.S. PRIME-LOW	U.S. T-BILL - 30 DAYS	U.S. T-BILL - 60 DAYS	U.S. T-BILL - 90 DAYS
JAN/81	20.12	19.93	15.15	15.44	15.93
FEB/81	19.42	18.97	14.55	15.36	15.52
MAR/81	17.84	17.66	13.73	13.86	14.01
APR/81	17.26	17.12	13.51	13.85	14.35
MAY/81	19.69	19.67	17.54	17.26	17.33
JUN/81	20.03	20.03	15.11	15.11	15.59
JUL/81	20.40	20.40	14.92	15.19	15.67
AUG/81	20.50	20.50	15.71	16.09	16.39
SEP/81	20.05	20.02	14.59	15.24	15.64
OCT/81	18.38	18.36	13.43	13.89	14.15
NOV/81	16.78	16.75	10.82	10.94	11.17
DEC/81	15.75	15.75	10.51	10.91	11.34
YEAR/81	18.85	18.76	14.13	14.43	14.76

The Direction of Public Data Bases at I.P. Sharp

In all but a few cases, I.P. Sharp Associates is not the originator of the data in the public data bases. We have no plans to become a huge data gathering organization. On the other hand, we are quite good at taking the data prepared by others, and by using our timesharing system and our computing expertise, acting as a data disseminator. The particular combination of skills and facilities at I.P. Sharp Associates will likely lead to a vast increase in the number and size of public data bases available on the system. For example, we are working on a new import/export data base known as the SITC data base which will likely occupy several billion bytes of on-line storage when complete. The source (and therefore, the organization responsible for data accuracy) will be the United Nations. The method of data access, the delivery, and the support of customers who want to use this data will be the responsibility of I.P. Sharp Associates.

In the world, there are currently many, many sources of data—but it is data in printed form. Journals, magazines, newsletters, and bulletins often contain combinations of text and tables of numeric data. I.P. Sharp Associates offers an excellent means of delivering the text and tables of data as they appear in printed form. Instead of waiting for days for the mail service to deliver a newsletter, the same information can be transmitted, as soon as it is published, via timesharing systems such as I.P. Sharp's. If the information is useful enough, users will pay the extra amount that it costs to deliver the text

in such a way. For example, the Weekly Statistical Bulletin of the American Petroleum Institute is prepared and officially released on the SHARP APL system each Wednesday afternoon. A sample of part of the Bulletin follows:

API WEEKLY STATISTICAL BULLETIN: *1
WEEK ENDED AUG. 20, 1982

U.S. REFINERY OPERATIONS
=====

(DAILY AVERAGES IN THOUSANDS OF 42-GALLON BARRELS)

DISTRICT	TOTAL REFINERY INPUT	CRUDE OIL RUNS TO STILLS	FOREIGN CRUDE RUNS	INPUT TO CRUDE STILLS	OPERABLE CAPACITY
EAST COAST	1,179	1,055	940	1,071	1,584
APPAL. NO. 1	92	89	--	89	162
DISTRICT 1 TOTAL	1,271	1,144	940	1,160	1,746
APPAL. NO. 2	3	1	--	3	66
IND., ILL., KY.	1,886	1,805	521	1,864	2,526
MINN., WIS., DAK.	279	275	129	275	293
OKLA., KAN., MO.	810	752	50	773	965
DISTRICT 2 TOTAL	2,978	2,833	700	2,915	3,850
INLAND TEXAS	542	480	30	517	616
TEXAS GULF COAST	3,065	2,659	948	2,704	4,304
LA. GULF COAST	2,035	1,835	589	1,936	2,785
N.L.A. AND ARK.	186	172	21	182	287
NEW MEXICO	79	74	--	77	120

With a minimum amount of knowledge, any user can interactively print part or all of the Bulletin on a terminal, rather than waiting for it to be delivered by mail. Since this API data can influence the production plans for crude oil refiners, many oil companies find it advantageous to receive the Bulletin as quickly as possible, using the I.P. Sharp system.

On the other hand, those who wish to *analyze* the data in the API Bulletin cannot do so when the numbers are part of a textual document. For that reason, the numeric portion of the API bulletin is available in the form of time series just like all of the other data bases available via MAGIC. This is quite clearly the direction in which many new data bases on the I.P. Sharp System will take, that is:

- a combination of textual data and a user-friendly keyword retrieval capability
- the capturing of the numeric portion of the text as numeric time series so that MAGIC can be employed for data analysis.

Summary

There are many companies selling data similar to the data in the I.P. Sharp public data bases, but it is the vehicle by which you can get at, manipulate, and present the data that can allow you to get the most out of that data. MAGIC, in an APL environment, is a language which is useful with minimum training to the novice, and flexible and powerful for the more advanced user. With a package such as MAGIC, you can truly let the numbers do the talking.

SUPPORT SYSTEMS FOR ASSET-LIABILITY MANAGEMENT

David W. Crouse
Manager, Financial Systems
Policy and Planning Division, Treasury Group
Canadian Imperial Bank of Commerce
Toronto, Ontario

The Canadian Imperial Bank of Commerce (CIBC) is one of Canada's largest banks with fiscal 1981 assets of 66.8 billion dollars and revenues of 9 billion dollars. CIBC has more than 1,600 branches throughout Canada. International operations are of major importance as reflected by our presence in 24 countries. The Bank employs some 36,000 people. CIBC's environment is characterized as one with: long standing tradition (the bank's roots go back more than 110 years); highly divisionalized organization; domestic and international banking (in addition to the 1,600 branches in Canada, the bank has branches, representative offices, agencies and subsidiaries in 24 countries); multiple businesses; and high volumes (CIBC has in excess of 6 million customer accounts and clears close to 2.5 million cheques daily).

Managing assets and liabilities for such a large financial institution is no mean feat. Success or failure of asset-liability management reflects directly on the bottom line and consequently has major impact on the financial well-being of the bank.

The Treasury Group has a global mandate to optimize the Bank's net interest margin and return on assets. In direct relation to this mandate is management of the Bank's cash position vis-a-vis reserve requirements with the Bank of Canada. Treasury runs the domestic money market operation of the Bank and has intimate liaison with the international money market operation with which it shares the same physical trading room. Within Treasury also exists the major responsibility for the determination of appropriate Bank policy and strategy as it impacts and is impacted by tax implications as well as the preparation of returns for our various domestic and international operations.

The role of our group, Financial Systems, within Treasury is to support the Group in terms of its decision-making activities. In particular, our responsibilities are three-fold:

- 1) provide data processing and systems support wherever necessary,
- 2) provide various information analyses on current portfolios, proposed products, etc.,
- 3) maintain and report from a repository of data relevant to the operations of Treasury.

In other words, our mandate is to provide a decision support function. One of the releases prepared by the sponsors of this conference stated "managers are paid to make decisions, and effective decision-making requires knowledge, judgement and information ... of the three, information is the element which exhibits the highest degree of unpredictability from one decision to another. The problem with information is that we either have too much of it or too little, or it is in the wrong place". Financial Systems must ensure that executive management has what they need when they need it. Our job is not to usurp their decision-making role, but to support it.

The paper will discuss support systems implemented to date and those planned as aids in providing senior management better information and tools with which to address the asset-liability question. Application of APL to these problems as well as the imposition of traditional data processing approaches and standards to an APL environment will be stressed.

THE FINANCE SIDE OR "WHO ARE THOSE GUYS ANYWAY?"

Asset-Liability Management

To put it simply, asset-liability management refers to the structuring of loans, deposits and other investment portfolios in such a way that the spread between interest earned and interest cost is maximized and, at the same time, the effect of interest fluctuation on this spread is reduced. This interest margin has a significant impact on the bank's overall performance as it translates directly into the bank's income before operating expenses, provisions for losses and income tax.

Historically, asset-liability management had been a relatively easy game to play as interest rates showed very little movement over time, and yield on investment assumed a proportional relationship with the term of investment. As a result, little attention was paid to the different term structure between loans and funding deposits. At that time, it would even pay to fund long term loans with short term deposits in order to capitalize on an imperfect market. However, current turns of events changed this picture drastically as interest rates began to soar in an unpredictable fashion, causing an inverse yield curve for medium to long term investments and an erratic pattern for short term investments. In such an environment, mismatch on term of loans with funding deposits in the wrong direction spells disaster. More attention is demanded to structure loans and deposits portfolios carefully to minimize such impact.

Optimization of the interest margin presents a different, though related, problem in asset-liability management. The traditional concept assumes all loans are funded from a liability "pool of funds" and, hence, imply a uniform cost of funds. This approach is questionable as it hides different characteristics of different loans and deposits portfolios. In order to facilitate the identification of vulnerable loan portfolios, it is desirable to be able to identify the source and applications of funds.

In summary, both objectives of asset-liability management lead to the break-up of liability portfolios in such a way that matching of loans with deposits/liabilities and source and application of funds can be identified.

In a complex business environment such as that of a bank, these asset-liability management functions are no easy tasks. Some general concerns are outlined as follows:

- 1) The two asset-liability management objectives may prescribe contradicting ways of dividing the liability portfolios.
- 2) Market conditions and existing business mix become the practical determining factors in the formulation of any matching strategy.
- 3) Due to the volume of business activities, it is a difficult task to provide timely information for the evaluation and monitoring of asset-liability management policies and strategies.

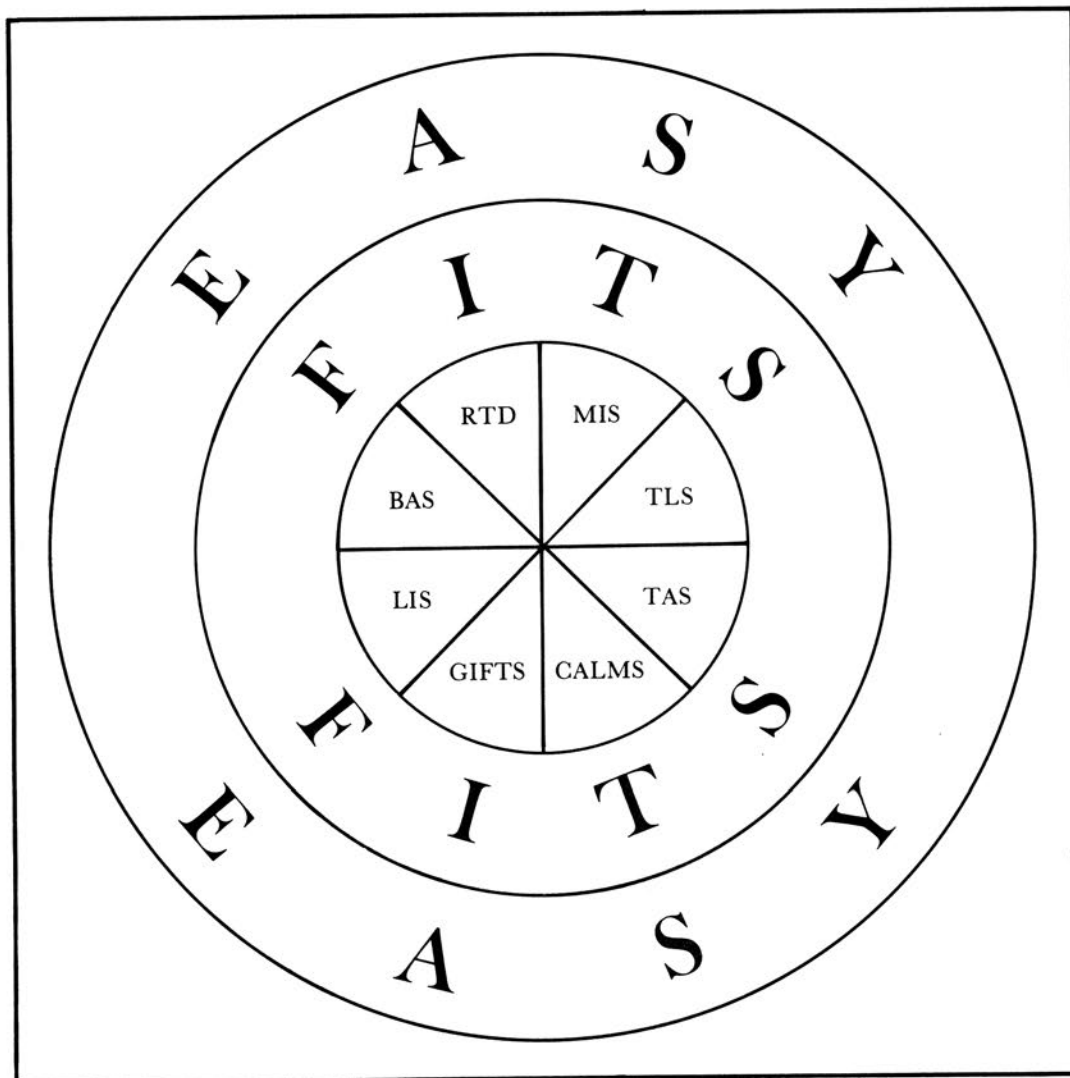
There is no easy answer to the first concern as it requires the discretion of management decision to choose one over the other. A comprehensive modelling package proves to be an invaluable tool to resolve the second problem. An optimal matching strategy often translates into an impractical one. It is more appropriate to analyze beforehand the result of hypothetical scenarios based on practical strategies before choosing one. Slight variations on the same strategy can also be fed into this modelling package to examine sensitivity of a strategy. The solution to the third concern lies somewhere between practicality and good data analytical skill. Depending on the requirement, detailed data may not be needed for all activities. For some instances, data in a summarized form may suffice. The trick is to determine what level of summarization is minimally sufficient to perform the required analyses. The problem of data volume can be reduced significantly by using summarized data.

Our systems are used to help determine the financial policy to be adopted by the Bank. They are designed to provide information on the Bank's current mix of assets and liabilities and its current financial position. Accumulated historical data is used together with current portfolio information, both in the evaluation of the current situation and for future-oriented analysis. The system has numerous levels of security and privacy to ensure the integrity and confidentiality of the information it provides.

However, before you can analyze data, you must have that data in the first place. Consequently we have had to design and implement operational systems collecting data from our branches as well as from many head office divisions. Sometimes we are able to plug into existing Bank computer systems and either scrape the data base or be provided with regular data which we subsequently massage and analyze to our own specifications. We feel the latter approach is the preferred one.

Given the mandate of Treasury, we are interested in all assets and liabilities of the Bank and have built systems around these categories. Our general approach has been to build a system for each asset/liability or type of asset/liability while at the same time being very cognisant of the need for bringing all the information together. The following diagram conceptualizes our philosophy in this regard:

Figure 1
Financial Systems



We currently have about eight major systems under way as represented by the acronyms BAS, CALMS, TAS, etc. in the centre of the diagram. Each of these is a self-sufficient information system, collecting data from source, manipulating and storing data, and generating operational reports which are distributed to all levels of Bank management, including executive management. These systems will be described later in the paper.

EASY (**E**xecutive **A**PL **S**Ystem) is a home-grown high-level management decision system consisting of reports and models to be accessed primarily by executive management. We currently have screens on the desks of senior management and they, through EASY, have on-line access to information from our production systems. Some of that data is reported in real-time as well. Regular production reports are not reproduced on EASY, but only condensed, requested information relevant to the decision-making responsibilities of the senior manager so involved. The basic design intent is for execu-

tive interaction with EASY. Executive management do not wish to become programmers or play games with a terminal. Our design reflects that and makes everything available at the push of a button—the buttons being, in this case, programmable function keys. EASY in turn accesses FITS, a time series data manager, which either has been updated by production systems or reaches into the production system for relevant data.

FITS (**F**inancial **I**nfomamation **T**ime **S**eries) is a home-grown data manager for data to be stored in a time series format for later analysis and reporting. It is updated in two ways. The first way is from the various production systems shown in Figure 1. Each system produces various reports and statistics some of which we wish to store in a convenient way for future use. Thus was FITS born. FITS is updated directly by our overnight processing wherever possible. The second way is by the manual inputting of data from hard-copy reports. FITS is extremely user-friendly and provides for the creation and updating of time series data via English-like prompts or requests. The user doesn't need to know the file structures being used; the "computer system" never appears.

I am going to discuss two of our production systems in detail because of the complexities and different approaches undertaken. The two are: GIFTS, an international funding system; and TD, a system to maintain term deposit information. The other six systems indicated in Figure 1 will be discussed more briefly.

GIFTS

GIFTS (**G**lobal **I**nternational **F**unding and **T**rading **S**ystem) has been designed with four major objectives in mind. These are:

- 1) Provide an accurate accounting system for all international lending, funding and trading operations
- 2) Provide up-to-date information on liquidity and maturity structure to facilitate the trading and funding functions of all international money market operations
- 3) Provide timely management information to facilitate decision-making regarding international funding strategy, to assess profitability for segregated trading portfolios, and to assess the trading efficiency of every international money market operation
- 4) Provide a financial modelling system to permit analysis of the expected outcomes of strategies for any asset and/or liability portfolio

The design of GIFTS follows the principle of decentralized operations in which each international trading unit (agency) has access to a stand-alone trading subsystem of its own to facilitate its trading function. Each trading unit is responsible for the operation of the subsystem including data entry, report retrieval, and on-line data inquiry. Information captured in each of these subsystems is accessed by regional offices and head office, either as individual or consolidated data sets. Transfer price effects for all intercompany transactions are eliminated in all consolidated data sets.

Assets are separated by portfolio type, rate structure, and term structure. Liabilities are segregated by funding requirement of the targeted asset portfolio. Data can be consolidated by as many as three levels of organizational structure: unit trading opera-

tion, international regions, and worldwide international operations (namely, Head Office). The first level is categorized as an operating unit while the latter two are referred to as consolidated units. An operating unit is one which has an actual trading function; a consolidated unit has no specific trading capability.

A financial modelling system is provided in GIFTS to enable analysis of the expected outcomes of strategy for individual asset portfolios. At the outset, the system will provide the ability to handle modelling for major fixed term assets such as LIBOR loans and Interbank placements. Outcomes of strategy expressed as amount, rate, issue period and maturity period are entered into the system in a special data entry routine. These outcomes can be stored, changed and deleted. They can be grouped in any combination for application to the actual portfolio. Results are produced in special modelling reports.

GIFTS is designed to be a global system through which international trading information is accessible by head office management. It provides essential operating support only to those units whose operations are not mechanized. For those units that are automated, the data entry function is replaced by system-to-system data transfer at regular intervals during a business day, whenever such transfer is possible. Only management reports are provided for these units.

GIFTS was developed and implemented in the SHARP APL environment. It was subsequently patriated when we moved in-house. The entire system operation is carried out by program-controlled dialogue between operators and the system. The I.P. Sharp international data communications network is used for worldwide system access via local dial-up. Equipment for a data entry station includes a CRT, a 1200-baud printer, a modem and a private telephone line. Equipment for automated operating units is determined by the mode of data transfer. Specially designed executive reports using data from GIFTS are made available to executive management via the EASY system.

Currently, we have operations in three major trading centres: Toronto, New York City and London, England. Data capture and collection are handled differently in each case.

In Toronto, a Bank system is in place collecting and capturing data relevant to trading operations. Each night, that production system is "scraped" and a sub-file of data for GIFTS is created, transported to the CIBC APL installation and subsequently processed by GIFTS. Reports are produced and distributed the next day.

In New York, operators sign on to the I.P. Sharp international communications network, load the appropriate GIFTS program, and enter transactions throughout the day. These transactions are batch-processed within a half-hour of the books being closed and reports are available thereafter.

London has their own data processing facility which has, among other systems, a trading system. Each night their production files are "scraped" and a sub-file of data for GIFTS is created on tape. That tape is sent by courier to I.P. Sharp's London office and transmitted via the I.P. Sharp network by an R-task to our in-house installation where it is processed. Reports are available by morning in Toronto and by mid-day in London. Future plans call for the London CPU to be directly linked to the I.P. Sharp network so that transmission of data can take place totally electronically.

TLS

TLS (**T**erm **L**iability **S**ystem) records data on wholesale term deposits; that is, term deposits of value \$200,000 or more. It is an operational system in that we (Treasury) capture data at source, update and maintain production files, and produce operational reports for the Bank. This is necessitated by our need for up-to-the-minute data on our position in these instruments—data which is not available by any other means.

When a customer wishes to take a wholesale term deposit, the branch will phone the trading desk for a rate to be offered to the customer. If the customer accepts the rate, the branch confirms that fact with the trader who then completes a data entry ticket and takes it to a data entry operator for keying into TLS. That information, upon entry, is thus made immediately available to the traders who have a terminal in the trading room as well as to senior management who have terminals on their desks. A major strength of TLS is that continuous, real-time information is made available to those people who must make decisions in the money market, a dynamic and fast-action environment.

TLS posts transactions throughout the day in a memo-system from which up-to-the-minute positions are reported. Data is entered from both Toronto and Vancouver trading centres. Overnight batch processing updates the master production files and produces reports which will be printed on high-speed printers in the data centre and delivered to operations staff upon arrival the following morning. Reports are also produced for branches and regional offices across the country. These reports are spooled onto tape and transmitted by courier to the Bank's major data centre where the reports are distributed via CIBC's communications network to the appropriate regional data centres.

Many reports are produced for TLS, one of our largest portfolios. These include: maturity schedules by date and/or by company; daily position, detailing structure and cost by security type; daily and weekly activity and new business reports; regional and national breakdowns; interest rate analyses; customer report of current and year-to-date business; and a generalized inquiry by several search fields including customer, term deposit, issue and/or maturity date, branch and amount.

RTD

RTD (**R**etail **T**erm **D**eposits) is the system for term deposits under \$200,000. We maintain separate computer systems for the two types of term deposits because of the significantly different ways these two deposit types are handled. They are booked differently, their information reaches us in different ways, and their impact on the pool of funds of policy decisions taken by the Bank differs. The major difference between TLS and RTD is that in RTD we deal only with summarized information whereas in TLS we run an operational system based on individual term deposits. Because there are so many individual deposits under \$200,000, because this bulk of money is not prone to change location quickly due to changing Bank policy, and because we are interested in cash management of a portfolio and not accounting for individual deposits, we decided to deal with a macroscopic view of this instrument. For example, if 25 term deposits at 18% are booked for 90 days across the country today totalling \$500,000, we receive one transaction indicating a value of \$500,000, issue date of today, term of 90 days, interest rate of 18%, and comprising 25 deals. We do not get particulars from those 25 deals. We don't need them.

Branches can take retail/term deposits without phoning head office for rates. They get rates from a rates bulletin circulated nationally upon any change in rates. Branches enter details of retail term deposits taken into on-line teller terminals connecting branches to our major computer centre. Every night, the day's transactions are summarized by issue date, maturity date, and interest rate and a special tape is created for Treasury. That tape is transmitted to our APL installation where we process it and create reports. Thus the position of this portfolio as seen by Treasury management is static throughout any business day. Types of reports include: maturity schedules; changes in position on a daily, weekly, and monthly basis; interest rate analyses; and new business and outstanding position statements.

BAS

BAS (Bankers' Acceptance System) monitors and reports on this financial instrument by which a customer is able to borrow funds in the money market with repayment guaranteed by CIBC. The major concerns here are the Bank's potential exposure in terms of total dollars authorized for customers to issue Bankers' Acceptances (BA's), and the impact on cash management of the Bank due to redeeming BA's upon maturity.

Information as to customer authorizations is collected from the three head office divisions which have authority to grant lines of credit through Bankers' Acceptances. Primary among these divisions is our Credit Division. Credit inspectors responsible for the client in question prepare a one-page authorization report whenever a line of credit is granted or modified and send it to our cash management function in Treasury. Cash Management personnel enter the data as received and consequently an up-to-date position is always available. Branches are required to report weekly on their position in BA's and these reports are verified on a cyclical basis against the computer system.

Data pertaining to actual customer availments is of major importance to us because of its impact on our cash position at any point in time. There are essentially two CIBC branches in Canada which redeem BA's upon maturity and to which all advices of BA's being issued must be sent. These branches are in Toronto and Montreal. Cash Management receives phone calls twice daily from these two redemption centres giving pertinent data on what has been issued thus far during business on that day. This data is entered immediately into the computer system. At the end of the day, the two redemption centres confirm their phone calls by memoranda which are subsequently verified against a computer-generated transaction log. Report requirements centre around: activity and maturity schedules for availments; regional and national analyses by company, by branch, by availments and by authorizations; and various ratios and statistical analyses on the portfolio itself.

LIS

LIS (Loan Information System) is being constructed so as to allow us to manage a portfolio which heretofore has been treated on an individual loan basis but which requires a macroscopic view because of its significant impact on both cash management and asset-liability management. Two approaches are evident in this system. In the first, we are accessing the Bank's computerized loan accounting system which is fed by all branches entering details of their loans through on-line teller terminals which are connected to our Bank's central computing facility. A copy of the production file is taken for us each week and set up for our access on a mainframe other than those CPU's housing active production systems. We then have on-line access to the data as well as the facility for executing batch jobs against the data.

Currently we are looking at demand loans and fixed rate loans with two objectives, namely analyzing foreign currency positions and yields, and determining corporate loans profitability by looking at outstanding position and yields. Having derived this set of data, we then compare against cost of funds, administrative costs, and margins to determine profitability. We will also be looking at borrowing patterns of major customers, particularly regarding their use of the money market so that we do not lose money on arbitrage.

The second approach we are utilizing is to go straight to the branch system in requesting data for certain types of loans in which advance notice of the customer's intent to borrow is available. In this way, by having data sent by facsimile transmission to head office from regional offices as loans are agreed with customers, we are able to forecast the impact on cash management and funding requirements ahead of actual drawdowns. Thus our ability to manage our cash and match assets with liabilities is greatly enhanced.

TAS

TAS (**Tradeable Assets System**) is a system designed to support the Bank's dealings in discounted financial instruments on the secondary market. These instruments are held as investments by the Bank and include Bankers' Acceptances, Bearer Deposit Notes, and Treasury Bills, both Government of Canada and provincial. Data is keyed in by our data entry personnel from source documents completed and sent by our trading desk. Reports centre on the outstanding position of our portfolio, analysis of profit on sales broken down into trading profit and interest profit, and daily transaction logs. Analysis is based on paper type, maturity date and issuer (that is, guarantor).

CALMS

CALMS (**Commerce Asset-Liability Management System**) is both a package of smaller systems as well as a data base manager. The package consists of programs to maintain the average account balances as well as earnings or costs associated with each category within the Bank's Statement of Earnings and Costs. In addition, we execute an analysis of our net interest margin comparing periods and identifying changes due to amount changes, interest changes, and the interaction between amount and interest changes. As a data base manager, CALMS maintains Bank-offered market rates such as CIBC prime rate, savings rates, etc. It also contains a section which maintains: a Bank holiday (that is non-business day) file; a branch file showing branch transits, names and regions; a trader file detailing numbers and names for each of our money market traders; and a customer file with names and Treasury-assigned numeric cross-references of customers with whom Treasury deals vis-a-vis money market instruments.

MIS

MIS (**Management Information System**) was originally conceived in response to a request by a vice-president to reduce the amount of paper entering his office every day. We subsequently undertook a paper and information flow analysis for that individual and determined, with much assistance from him, what particular numbers or data he was really looking at and for what purpose. We were then able to reduce and reformat the information he wanted to see into about seven screens. Subsequently a system was designed to intercept the paper going to this individual, capture required data, and manipulate it for presentation within the EASY system. The data is all stored in FITS and is updated either directly by the respective production system or from manual data entry.

THE HARD SIDE OR "NUTS AND BOLTS AND A LITTLE GLUE"

Hardware

We use SHARP APL almost exclusively for our development and analysis activities. A few old and very small systems are written in Fortran and running on the FRI time-sharing service bureau. Until July 1, 1982 we accessed the I.P. Sharp time-sharing service bureau but due to our ever-increasing use of this facility we decided, about one year ago, to go in-house with SHARP APL. The past year was spent in negotiations and planning among ourselves, I.P. Sharp, IBM and other suppliers, and our Systems Division who lease and operate the equipment and software on a charge-back basis to Treasury.

Our central processing unit (CPU) is a stand-alone, APL-dedicated IBM 4341-2 with 8 megabytes of main memory and 6 channels. We have an additional IBM 4341-2 available to us as a backup machine in the event of prolonged failure on the designated host. The system control program is IBM's MVS/SP 1.3. Additional IBM software are HSM (Hierarchical Storage Manager—a disk management system for backing up data files) and RMF (Resource Management Facility—a monitoring and reporting tool running within the system for analyzing I/O and CPU performance and utilization). ACF2 (Access Control Facility—a data set security package) is being installed on other systems and when proven, may be installed on the APL CPU. Other software are, of course, the SHARP APL distributed product, and OMEGAMON, a package utilized in real-time to provide an overall, quick picture of system and resource utilization.

The hardware configuration supporting the 4341-2 includes 10 dedicated STC 8360 DASD spindles and shared access to others, shared access to 12 STC 8600 6250 bpi tape drives, 2 IBM 3278's and 1 HDS Concept APL/4 as local consoles, and an IBM 3705 communications controller which functions as the network link (node in the I.P. Sharp international communications network). Current workload profile is 24 terminals (18 in Toronto, Canada; 2 in Vancouver, Canada; 3 in New York City, U.S.A.; 1 in London, England) with approximately 40 concurrent tasks. Maximum user load contracted for with I.P. Sharp is 50 at this point in time. The computer centre is located outside downtown Toronto.

Access to the APL system is through dedicated lines, dial-up lines, and the I.P. Sharp world-wide network from Canada, United States and England. Our terminals include approximately 14 HDS Concept APL/8 display screens, 10 LA120 printer terminals, 5 Texas Instruments Silent 700's, and a few AJ832, AJ510, LA34, LS400/4 terminals. Access to the APL machine is controlled by the IBM 3705 communications controller which can be entered via one of the following three methods:

- 1) Direct Lines—for those terminals in Head Office, Toronto. Our terminals are connected to Gandalf LDS120 short haul or limited distance modems capable of 1200 baud transmission. These in turn are connected to a Timeplex M24C Statistical Multiplexor capable of handling 24 terminals but initially set-up for a load of 16. The multiplexor is back-ended with two Gandalf SM9600 "super modems", each connected to a dedicated data circuit operating at 9600 baud transmission. Computer Communications Group (Bell Canada) provides one line and CN/CP Telecommunications provides the other. Both circuits operate simultaneously with one line automatically taking over in the event of failure in the other line. Each circuit thus connects Head Office in downtown Toronto with our computer centre outside of town. At the computer centre, these circuits run into two other

SM9600's connected to another M24C multiplexor which finally connects to the 3705.

- 2) Dial-Up to CIBC Machine—for terminals with couplers, data sets or modems having local (to Toronto) telephone dial-up capability. Terminals are connected to acoustic couplers (AJ242A for 300 baud and AJ1234 for 1200 baud, or built in as in the case of the Silent 700's), Bell-provided data sets, or modems (Vadic 3451 SSX Triple Modem—switch-selectable for either 300 or 1200 baud but within 1200 baud intelligent enough to transmit in either Bell 212 or Vadic protocol). Connection is then made through normal voice grade telephone lines to the computer centre where Vadic modems receive the calls. There are 5 VA3467 modems which can receive either 1200 baud or 300 baud calls. These five modems then connect to the 3705.
- 3) Dial-Up to I.P. Sharp Network—for terminals outside of local-to-Toronto telephone dialing or in the event of multiplexor failure in Head Office. Terminals are connected to the same couplers as discussed in 2 but in this case the phone number called is that of nearest local access to the I.P. Sharp network instead of the Bank's phone. The I.P. Sharp network passes the data to Toronto and through to a Gandalf SM9600 modem resident in the I.P. Sharp computer centre. This modem connects to a dedicated 9600 baud data circuit provided by the Computer Communications Group (Bell Canada) running from the I.P. Sharp computer centre in downtown Toronto to the CIBC computer centre. At our computer centre the line feeds into a SM9600 modem and into the 3705.

The three foregoing methods of accessing our computer are shown in Figure 2, "Access to CIBC APL Machine". Figure 3, "APL Workload", displays the projected workload on the 4341-2 based on current and short term plans of Treasury.

Figure 2

Access to CIBC APL Machine

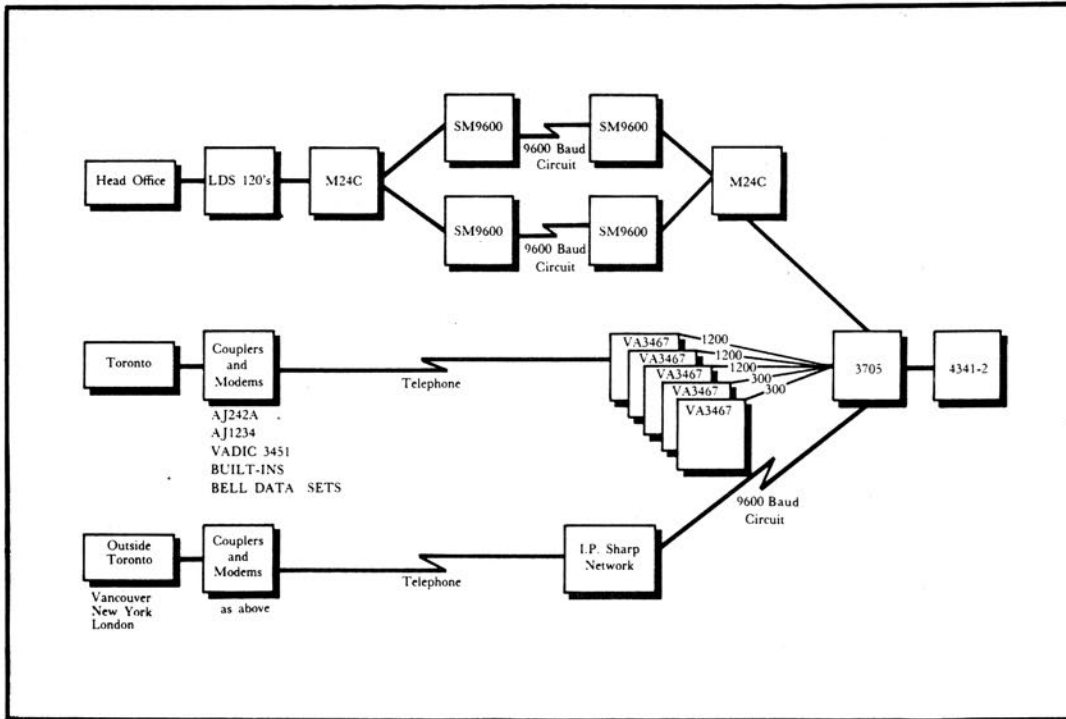
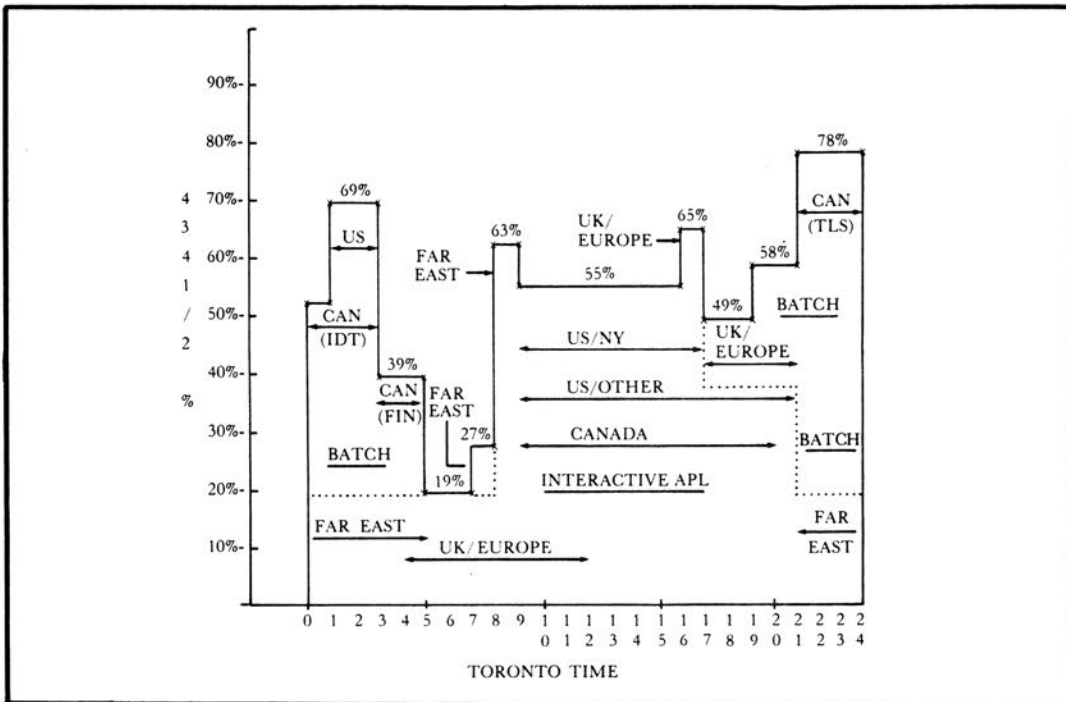


Figure 3

APL Workload



Service is available 7 days a week on a 24-hour basis except for Saturday from 14:00 to 24:00 when scheduled downtime exists for backups and maintenance. When possible, emergency maintenance will be scheduled for 05:00 to 07:00 on that day. All users are notified in advance of such service outages, whenever practical. As mentioned before, another 4341-2 processor is available as backup during extensive outages. As a third-level worstcase backup, tapes of our workspaces and files would be taken from our computer centre to I.P. Sharp and we could then execute high priority systems there.

Software

We use a number of SHARP APL facilities in order to fulfill our mandate to capture, analyze and report on data for asset-liability management. This section examines use of background tasks and shared variables for capturing data, maintaining data bases, and creating report files. We also look at a special application of share-tasks which we tested during the user acceptance phase of our in-house migration.

Data capture systems

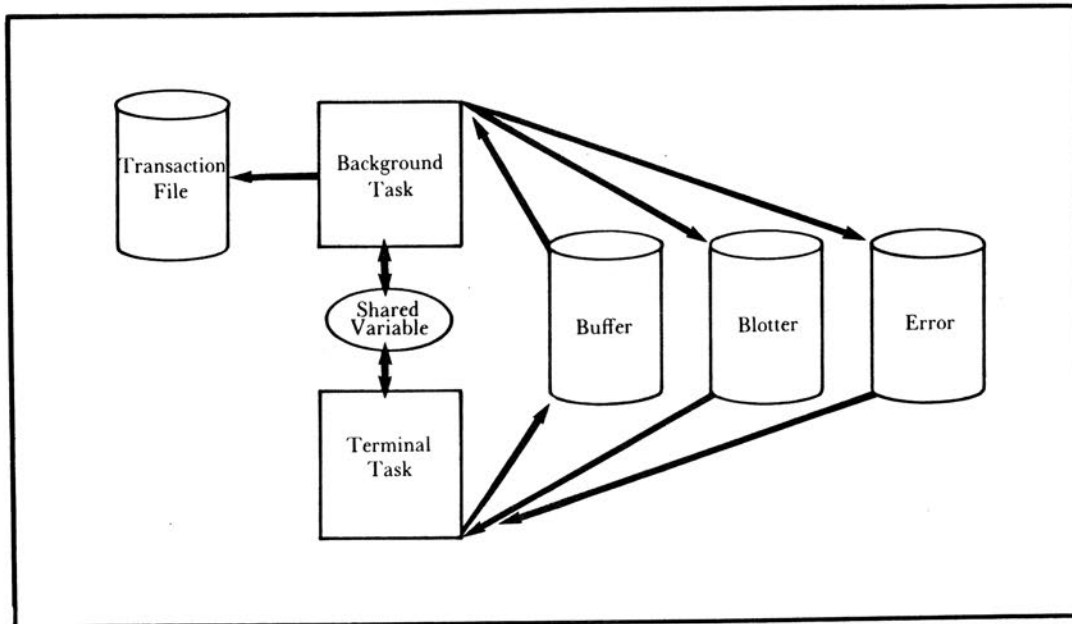
Treasury maintains a number of data bases by capturing, through interactive APL input procedures, entries for each transaction. The number of transactions entered for these data bases vary from a few per week to almost a thousand per day. For most of the data bases, these transactions are stored on a transaction file for later processing (see next section).

The larger systems allow for two modes of input: interactive, which uses only a terminal task; and deferred, which uses a background task. The interactive mode prompts the user for each field on a transaction ticket, and does whatever checking is necessary immediately before updating the daily transaction file. This mode is particularly helpful for educating inexperienced operators.

In deferred mode, the user is asked to enter each ticket as one response, with each field separated by an appropriate delimiter. In this mode, these responses are stored in a buffer file for processing and stored into the daily transaction file after the user enters *)END*. At this point the terminal task signals, via a shared variable, a background task that data is available for processing. This background task now processes these transactions, verifying data inputs, and stores either a blotter (that is, a formatted version of the inputs) in a blotter file for accepted entries or an error report in the error file, both for later printing. After storing a blotter, the background task will signal via the shared variable that there is data in the blotter file to be printed. In effect, this allows the terminal task to be printing these blotters while the background task processes more entries. After the blotter file has been printed, the error file is printed. The interaction between the terminal task and its background task is shown in Figure 4.

Figure 4

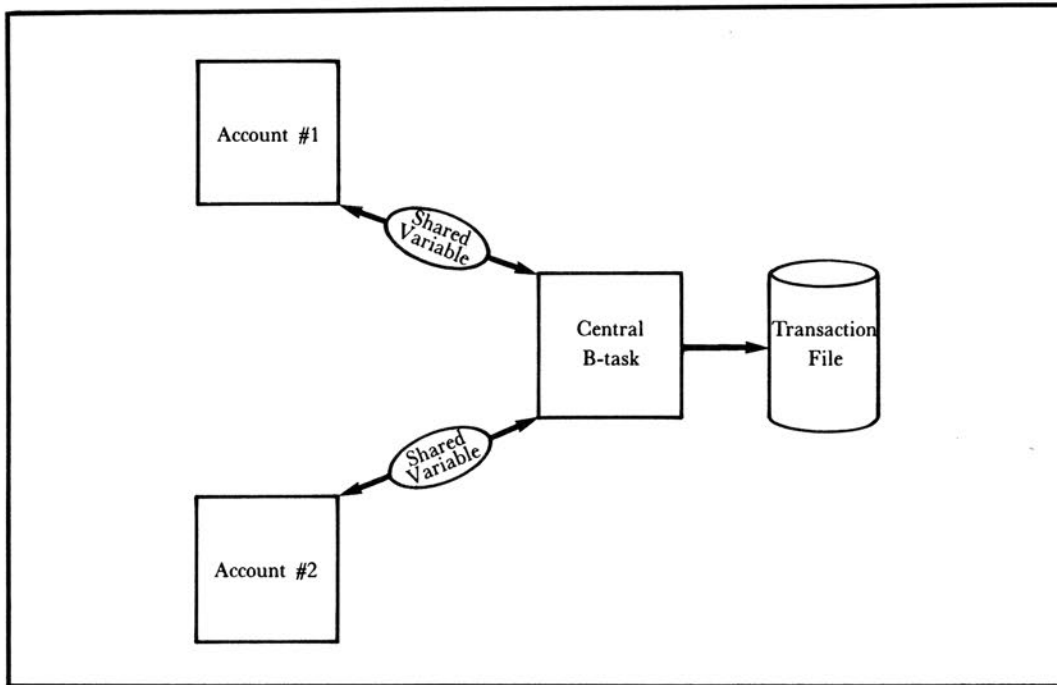
**Deferred Mode Input: Interaction
Between T-Task and Background Task**



Deferred mode allows for the bulk of the processing to be handled by a background task, which reduces costs of data entry. The one prompt per ticket also saves CPU usage and character transmission.

Most of the larger systems allow for more than one user to enter data at the same time. This is done by one of two procedures: a simple quad-hold on the file being updated; or a central background task using shared variables to communicate with each input task and to update the file. In the latter case the input task may be either a terminal task (interactive mode) or another background task (deferred mode). Figure 5 depicts the use of tasks for multiple input users.

Figure 5
Multiple User Input



This dependence on shared variables in a large daily system has educated us in the use and misuse of shared variables, and some of their idiosyncracies. One source of problems when using shared variables to communicate with a background task occurs when the background task bombs, either on a shared variable interrupt or, heaven forbid, a programming error. In this case the shared variable reverts to a normal variable, with the additional implication that using and setting of this variable is no longer controlled by the shared variable processor. For example, one could use this shared variable to inquire about the name associated with a customer number. As long as the background task works properly one will get the desired result. However if the background task fails, the variable does not get assigned the appropriate name. Instead, it still is the customer number which we originally assigned to it. Under certain conditions this may lead to undesirable consequences. The user should not automatically assume that the shared variable is operating as expected. A procedure must be in place to examine the results for correctness and proper action must be taken when there is a problem.

Batch processing of transactions

We found that except for certain management information requirements, most reports and the actual updating of data bases need not be completed until the next day. This has allowed us to use batch tasks to take the transaction files for the day—either as entered by our operators or created from files from other sources—and do the bulk of the processing. This processing includes maintaining our outstanding and historical data files, reports, and pre-calculated information for on-line reports. These batch tasks allow us to distribute the processing load more evenly on our machine.

Many of our recent systems now include files which contain reports. These files contain physical copies of reports and can be accessed individually or as a group. This allows for users in different locations to access the same reports without re-calculation. This principle also applies to the pre-calculation of data needed for on-line management reports. These often are stated in terms of closing position from the previous day plus changes today.

Share task application

While we do not make regular use of share tasks, we found it useful during our user acceptance testing of our in-house system. The problem was to confirm that all of the 500 files and 200 workspaces on our I.P. Sharp accounts that we expected to be transferred, in fact were transferred. By using the share task facility we were able to sign on to each of our 50 accounts and store a list of all files and workspaces in one place. This was done on both the I.P. Sharp computer before transfer, and on our machine after transfer. These two lists were then compared to ensure proper transferral.

THE SOFT SIDE OR “WHY WE THINK IT WORKS”

Our approach

APL is not simply a programming language—it is a philosophy! It is an approach, a concept whereby responsive, flexible information systems can be implemented quickly without sacrificing accuracy, data integrity, or security. Paradoxically, or so it seems, the imposition of some traditional data processing approaches and standards can make the APL-based environment a reliable, on-going support facility. We believe our approach has blended many of the advantages of both philosophies and want to share it with you. I cannot stress enough the need to function as an on-going entity. Too often I have seen APL support efforts function as though everything was a quick and dirty one-off and required no planning, documentation or consideration vis-a-vis interaction with other data or “one-off’s”. Quick and dirty one-off’s aren’t! If it was needed once, chances are almost perfect that it will be needed again—plan it that way from the outset and you will save yourself time and grief later on. If you are perceived to be a “one-shot” support group, you almost never will be given the opportunity to be anything else, and consequently will be doing yourself and your organization an injustice. The following discussion is presented in what I feel is priority sequence. However, I must stress that all of the following areas are important.

Know your business

We support the Treasury Group of the Canadian Imperial Bank of Commerce. Our company's business is banking; our group's business is finance—the management of the Bank's assets and liabilities. Therefore my business (read Financial Systems business) is finance! My second business is banking. Only in third place is my business—computer systems, APL, and the like. If you do not understand the business you are supposed to support, you will never really support it. You have to know what is important to your management. Don't wait for them to tell you what they need or how they want it—that, unfortunately, is a philosophy adopted by many traditional data processing functions in various organizations. You have to appreciate the peculiarities of the business, the subtle nuances, the unique time frames, pressures, and so on. To do less only limits the contribution you must be able to make.

Support your management

You are in business for the convenience and assistance of senior management and not—surprise, surprise—vice versa. Make it easy for the user of any system you build, even if the programmer will have to go through endless gyrations to make it happen. Never tell someone “it can't be done”. How many times have you heard an analyst say those very words when the requestor had barely finished talking? Analyze the request. What is the person really needing? What is required in order to provide that? How much time and resources will be required? Go back to the requestor with this information so he/she will be in a position to make the go/no-go decision. Don't complain or moan because of the extra work involved—that is exactly why you are in business in the first place. You are in business because your management is in business!

Respond when necessary

To be of any value, data is only useful when it is timely. If your management has a meeting at 10:00 Friday morning for which he needs a particular analysis, it does him and you no good if the report isn't ready by then. Your value to the organization will depend on your ability to respond when and where needed, even if it means the midnight train home once in a while.

Human factors

Your senior management—the people you are supposed to be supporting—are where they are because they are good at their business. Do not expect them to be good at yours; that is why they hired you in the first place! Design your systems, your reports, your terminal dialogue and interactions with that in mind. Provide manuals for them in basic English, not APL-ese or some such technical dialect. Avoid using your jargon in discussion or reports with those you support and interact.

Team effort all the way

We function as a team in Financial Systems. Managers are not automatically better or smarter by virtue of their positions alone. All pertinent views are sought and considered. Only then are decisions taken. We expect staff to receive criticism and advice from each other, as it is intended, that is, constructively. Everyone must be

prepared to recognize that only by functioning as a team can we achieve what is required. Consequently we have no use for the APL superstar or the backroom rocket scientist to whom interruption and team play are unwelcome.

We also employ the team concept in rotating personnel among systems so that more than one person is familiar with each system. In addition to system development involvement, each staff member is required to carry some maintenance systems as well. This increases their knowledge of the department's activities as well as providing several layers of backup.

The other aspect of team play is that we are part of the Treasury team, as well as the Bank team, and consequently must be aware of and play by the "play-books" of these entities as well. That is, we are part of an organization with its own procedures, rituals, communication, etc., and therefore must expend time and effort in learning how to participate in this arena as well. As an aside, if the group you support works 9-5, then your people had better be available during those hours as well. Work at least those regular hours—midnight specials with mornings for squash just doesn't cut it.

Project start to finish

The same person/people is/are responsible for all phases of a project—interviewing potential users, analysis of requirements, system design, programming, testing, documentation and implementation. We do not have someone do the analysis and design and then hand over specifications to a programmer. Our belief is that the user will get what she wanted if the same people are involved throughout. We are not interested in the analyst who doesn't want to get his hands dirty at the terminal or the programmer who doesn't want to talk to the user.

K.I.S.S.

The object is **not** to build the best, the most complex or the most sophisticated system the world has ever seen. Effectiveness is the order of the day. In most cases we deal with, it is not important that a job or system was done the best way possible—only that it was done! That is not to say we don't care about efficiency or sophistication, but that this is not the end in and of itself. The overriding philosophy is Keep it Simple, Stupid.

Where's your road map?

You wouldn't think of leaving home on a major trip not knowing where you're going or how you're going to get there. So why do it when you build systems? Despite the allure of designing at the terminal, we don't allow our people to touch the machine until they show they have an understanding of the problem, the solution, and the path they are going to take from the former to the latter.

To that end, we insist that a Requirements Document plus other standard papers be prepared prior to terminal work. The Requirements Document, prepared after interviewing potential users and analyzing the problem, contains three basic sections:

- Reports - this consists (for every proposed report) of a sample report layout and a narrative section discussing report contents, time and frequency of production, distribution, and notes relevant to that report.
- Data - a 3-part section discussing data to be collected as input to the system, data to be derived during processing, and data to be accessed from other systems.
- Procedures - procedures and responsibilities surrounding the proposed computer system.

Other standard papers include proposed file structures, prompting sequences (for on-line interaction), edit/validity/integrity checks and assumptions, system flow (depicting the relationship between files, processes and output) and a macro program flow. This is done prior to terminal work. It is not wasted effort. Most of your documentation is complete before you start, you know where you're going, and if someone has to be brought into the project for whatever reason, continuity of effort is assured. Remember, if you can't put it on paper, then you don't know what you're doing.

Top down

Design, programming, and implementation are all done this way. If you are not familiar with the concept, find out. The approach facilitates phased implementation, logical integrity and system flexibility.

Standards

Yes, we have them. The following discussion is meant to highlight some areas but is not meant to be taken as exhaustive.

- Programming - program and variable names should be indicative of their function; e.g., *GETNAME*. First line or more are to be comments outlining purpose of the function. Each function is to have one purpose. If a function does more than one thing, break it up. No function may be more than 60 lines long (so as to be printed on one sheet of paper) at the maximum and preferably should be less than 23 lines (for terminal display).
- Documentation - files and programs are to be documented on forms designed for that purpose. Component 1 of any file is reserved for the description of that file.
- Utility Functions - commonly used functions are shared in a utility file for use by all of the personnel. Not only do they save time in programming, but system maintenance is eased by the recognition of utility names and purposes. These utilities are treated as "black-boxes" with required inputs and expected output documented, but not the process.

- Restartability - all programs must be automatically restartable.
- Security - we employ several levels of security within systems for our files and programs.

Project management

The management team (four members) in Financial Systems meet once every two weeks for approximately 1-2 hours to discuss current as well as future work viewed both by project and by personnel. Project assignment and priorities are set for the next two weeks. A formal project report reflecting those decisions is distributed to the management team within two days of the meeting.

Staff meetings

Staff meetings are held every Friday afternoon for about two hours and all members of Financial Systems attend. The first hour or so is in seminar format where an invited speaker, usually from within the Bank, will speak on some aspect of banking or finance discussing the mandate of that division/department, how it goes about its business, and the interactions it has with the Treasury group. Other seminars will feature one of the members of Financial Systems discussing a particular system aspect or technical idea which would be of interest to other members of the department. These seminars have been very successful in introducing department members to other facets of the Bank as well as allowing our staff to meet people about whom they have heard and with whom they may have future dealings.

The second part of our Friday meeting is devoted to a round table discussion on the "business of the week". Department management take the opportunity to discuss events and policies to which they have been party and which staff might not have come across in their normal day-to-day activities. Each member of the meeting then discusses what they have been working on that week and plans for the following week. This provides a good understanding of what each other is doing, allowing for good interaction and help among the staff. We feel it contributes to a general feeling of contribution and belonging. We have little patience for and consequently do not subscribe to the "mushroom theory of management" wherein staff members are kept in the dark, and fed a lot of manure.

Time sheets

Yes, we do use time sheets. No, I do not liken our environment to the shop floor. When time sheets were implemented, we emphasized that we were not trying to keep tabs on people and whether they were working 37 or 39 hours per week. The intent of time sheets was two-fold. I wanted the people to be conscious of time in and of itself, and to recognize time as a valuable resource. In particular, I wanted them to be aware of where their time was going. Our environment is a "respond when requested" situation and we are subject to unpredictable requests for data, hardware assistance, and policy discussion on an ad-hoc basis. In such an environment, it is crucial to monitor time so both the individual and management know what is being executed and the resources required.

Information derived helps me in planning current workloads and in predicting time frames within which future work is to be accomplished. It also helps me in indicating relevant "soft" costs to senior management of work undertaken both for our own Treasury Group as well as other divisions in the Bank. In the latter case, this information is the basis for costs to be charged back to those divisions for work done at their behest. Time sheets are completed daily and submitted weekly to up-line management.

ACKNOWLEDGEMENTS

The author gratefully wishes to acknowledge the contribution of Gary Yee and John Vanderzwet of the Canadian Imperial Bank of Commerce in the preparation of this paper. Indeed, without these two gentlemen, much of what has been discussed would not be in existence at the Bank today. In addition, appreciation goes to Noulia Tzogas of CIBC and Rosanne Wild of I.P. Sharp Associates for their assistance in the typing and publishing support of this manuscript.

A PERSONAL HISTORY OF APL

Michael Montalbano
APL Systems Development
Santa Teresa Laboratory, IBM Corporation
San Jose, California

Introduction

I have several reasons for calling this talk a **personal** history.

For one, I want to make it clear that the opinions I express are my own; they are not the opinions of my employer or of any other organization, group or person. If you agree with them, I am happy to have your concurrence; if you disagree, I'd be happy to defend them. In any event, the praise, blame or indifference my views may inspire in you should be directed to me and to no one else.

What I plan to discuss are things I have done, seen or experienced at first hand. Thus, this talk is merely an opinionated collection of anecdotes. I want to emphasize this from the outset; it is my second reason for calling this a personal history.

But my most important reason is that I feel strongly that we need a good, thoughtful, accurate history both of APL and of computing itself. While I'd be flattered to have this account included as part of that history, I don't want anyone to mistake it as an attempt at the real thing.

The importance of history

We neglect history at our peril. The truly incredible growth of digital computer technology has transformed our world almost overnight. This transformation is not only continuing, it is accelerating. It gives every promise of continuing to change our institutions and the circumstances of our daily lives at a faster and faster rate. If we are ever to understand where we're going, it's important that we take a long, careful look at where we've been, where we are, and how we got from there to here. If we don't do this, we won't be able to control events; they will control us. As far as I'm concerned, that's what is happening right now; a runaway technology has us at its mercy because we have not developed techniques to understand and control it.

H.G. Wells described human history as a race between education and catastrophe. This observation is more pertinent today than it ever has been in the past. But, in the specific case of the proliferation of stored-program digital computers, the education-catastrophe

race, in my opinion, takes a particular form: it is a race between **technology** and **methodology**, between **gadgets** and **ideas**.

We are developing gadgets at an explosive, accelerating, self-fueling rate; we shall be swamped by these gadgets if we don't hasten to develop and apply the ideas we need to control them.

To me, this need defines the importance and the mission of APL. APL provides the best set of gadget-understanding and gadget-controlling ideas currently available. Looking to the future, it provides the best base for the methodology we **must** develop if we are ever to bring our gadget-based technology under control.

This opinion is based on experience. I was working with computers when they were merely gleams in the eyes of the early designers; I was working with APL when it was nothing but a collection of incomprehensible characters scattered through publications with catchy titles like *The Description of Finite Sequential Processes*. In other words, I've been working with both computers and APL since their very early days. And, in looking back on this experience of just over 34 years (which is my own personal experience of computing), I find that it can best be summarized in terms of two key ideas:

The **stored-program** idea: the idea that a procedure or algorithm can be stored as a collection of switch settings in exactly the same way as the data on which the procedure is to work and, as a consequence, that executing such a stored procedure consists of starting it and letting it flip switches until it is done. This was apparently first stated explicitly in a paper drafted by John von Neumann in 1945 [Ref. 1].

The **efficient-notation** idea: the idea that these vast collections of switches, changing their settings in thousandths, millionths, billionths, trillionths,... of a second, would be pretty hard to manage if we didn't develop a good way to describe them and think about them. I don't know whether this idea was ever presented anywhere in just these terms. Instead, it seemed to be implicit in the activities and writings of many people. But, in my opinion, it received its most effective and fruitful expression in the writings of Kenneth Iverson and others who followed his lead. The most important publication expounding this idea is the book from whose title the letters APL derive their significance [Ref. 2].

The **stored-program** idea provided, and continues to provide, the basis for our current runaway technology. The **efficient-notation** idea, if we take it seriously and do a lot of thinking and hard work, will help us curb the runaway and direct it into fruitful and productive channels.

You may find this brief summary of computing history controversial. I hope so. We need vigorous, informed philosophical controversy. And, as a contribution to this controversy, let me state some of my other biases in as controversial a manner as I can.

Biases of an APL bigot

There exists a growing class of people, of which I am happy to consider myself a member, who are called "APL bigots" by friends and foes alike. The friends use the term affectionately; the others do not.

The bigotry consists in believing that APL is the way computing should be done. I think it's fair to say that no one can properly be called an APL bigot who doesn't believe this.

In one respect, my bigotry may not be as great as that of the general run. I believe that **APL and assembler language** are the way computing should be done. To this extent, I am held suspect by true APL evangelicals.

In another respect, my bigotry is so much greater than theirs that, now that I am making it known, I fear I may be excommunicated as a schismatic.

I believe that, important as APL is in computing, **it is even more important as an instrument for rationalizing the management process.**

Modern management is in trouble. Don't take my word for it. Read the daily papers, the weekly newsmagazines and the flood of books and articles that describe management's difficulties and offer cures. Some of these, of course, assert that Japanese management is an exception to the general rule; the cure these people offer to non-Japanese management is to learn from the Japanese how to do it right.

I don't believe this. I believe Japanese management is in as much trouble as any other. Its current successes are no indication that its management isn't afflicted by the same difficulty as the management of organizations anywhere else in the world. Expressed simply, this difficulty is:

Nobody knows what's going on.

Your reaction to this assertion may be emotional. You may believe it passionately or reject it passionately. If you have either of these reactions, you may not have understood what I said, so I'll repeat it:

The problems of modern management are primarily attributable to one cause:

Nobody knows what's going on.

So there you have it: the basic message that motivates this talk:

- Management's primary difficulty is that it has no good way to describe its processes and thus develop objective means to correct or improve them.
- Other fields have faced this problem in the past. The ones that have solved it most effectively are those that have developed a notation appropriate to their subject matter. The most conspicuously successful examples of notations that have virtually created the fields they describe are, of course, the specialized symbologies of the so-called "hard sciences".
- The workings of management can be expressed as extensive, intricate digital procedures. These procedures cannot be designed, analyzed or described effectively without a notation specifically designed for the purpose.
- **APL is such a notation.**

There are, of course, many other important difficulties that management faces: obsolescent plant, intensified competition, environmental concerns, employee morale, strained

labor relations, residues of past mismanagement (reflected in such things as excessive debt, inadequate capital, incompetents in key positions) and so on. But these difficulties are made unmanageable by our inability to describe them in a way that promotes insight and facilitates communication.

Clearly, in expressing this opinion, I am swimming against the tide. The widespread current belief is that we are experiencing an “information explosion”. How can I accuse management of being inadequately informed when virtually everyone else says that they are **over**informed? How can I say that management doesn’t have enough information when everyone else says they have so much information that they are swamped by it?

Let me be blunt: I think the “information explosion” is a myth. I do not deny—how can **anyone** deny—that we are generating vast quantities of paper, tapes, disks, etc., with optical or magnetic symbols recorded on their surfaces. What I **am** denying is that these recorded symbols, in themselves, are information. They are not. They become information only if, as a minimum:

- They are accurately calculated.
- They present a true picture of reality.
- They are understood by the person to whom they are presented.

Of these three conditions, only the first stands a better than even chance of being satisfied. For the purposes of effective management, all too much recorded data does not present a true picture of reality and is not understood by the person it is supposed to inform.

What does this all have to do with APL? Let me answer by recounting the pre-APL experiences (at the U.S. National Bureau of Standards, the U.S. Naval Research Laboratory, and the Kaiser Steel Corporation) that convinced me that the computing field needed, more than anything else, an efficient notation for describing digital procedures. Then let me follow up with the post-APL experiences (at IBM and Stanford) that convinced me that APL was the notation we needed.

The U.S. National Bureau of Standards, 1948-1952

I started to work in May, 1948, as a mathematician in the Applied Mathematics Laboratories of the National Bureau of Standards in Washington, D.C. I was given a job description which, like every other one I have ever had, bore no relation whatsoever to what I actually did. My first assignment was to program a division subroutine for the UNIVAC. The UNIVAC that was then being designed by the Eckert-Mauchly Corporation could add, subtract and multiply but it could not divide. (Division was added later. The list of computer instructions or “order codes”, to use the terminology of the times, was updated and given an identifying C-number as the UNIVAC’s design progressed. If my memory’s not playing me tricks, the order-code list that was current when I started was C-7).

I was given a description of the UNIVAC. It described acoustic delay lines, excess-three notation and end-around carry. I had the feeling that I was entering a dark, eerie world in which words would be used as charms and incantations rather than to communicate definite meanings. I was right.

I was surprised that no one took the trouble to show me the UNIVAC for which I was to program division. After a few days I learned why. The machine described so confidently and completely in the literature that I had been given had not yet been built. (The first UNIVAC was not delivered until three years later. But at least it **was** delivered. Our group spent a lot of time programming what were then called "feasibility tests" for a lot of machines that never got off the drawing boards.)

Programming for nonexistent machines started to pall after a few months. Our group had punched-card equipment (including the 602 calculating punch) with which we calculated mathematical tables. I switched to this activity at just about the time the Office of the Air Comptroller asked the Bureau of Standards for assistance in using electromechanical (later electronic) equipment to calculate Air Force budgets. With the procedures then in use, it was taking eighteen months to prepare a yearly budget. There was general agreement that this was not satisfactory.

I was assigned the task of getting budget computation mechanized under the direction of George Dantzig, who was in charge of the Air Force budget project. His job was to devise the calculations we were required to perform, that is, he told me what was needed. I wired the plugboards and, later, wrote the programs that gave him what he specified.

The original calculations were called "triangular model" calculations (I understand they were given the acronym TRIM after I left the project). The later calculations were solutions of linear-programming problems, applying the simplex technique that George Dantzig had originated. The name of the Air Force project, SCOOP (Scientific Computation Of Optimum Programs), like TRIM, suggests that wherever a computer goes an acronym's sure to follow.

I programmed the triangular-model calculations for the 602, the 602A, the 604 (the **electronic** calculating punch) and was about to program them for the CPC (the Card-Programmed Calculator) when the SEAC became available and I switched back to programming instead of plugboard wiring.

The SEAC (National Bureau of Standards Eastern Automatic Computer) was the first stored-program digital computer to operate successfully in the United States. (The first stored-program computer to run successfully anywhere in the world was the EDSAC, designed and built by the University Mathematical Laboratory at Cambridge, England.) I introduce this information here because, computing history being what it is, it does not seem to be available anywhere else.

Of the many memories and ideas derived from my four years at the Bureau of Standards, three are relevant to our present purpose:

- 1) **I learned to be cautious about how I used the solutions of large systems of equations with uncertain coefficients.** I believe the triangular model is an extremely effective, and neglected, tool for a large class of planning problems. But, if the coefficients used in its equations have a large measure of variability or uncertainty, you must use its answers with caution.
- 2) **I learned how desperately we needed a good notation for describing algorithms.** I sat through many presentations and discussions of solution techniques for linear-programming problems. These presentations were largely chalk dust and arm-waving. The essential ideas, which are extremely simple once they are understood, were obscured rather than illuminated by the terminology and notation used to describe them.

- 3) **I learned that gadget development was outstripping idea development and would continue to do so unless we did something about it.** I suggested that we do something about it. My management was receptive to the idea (or told me it was) but said there was no money in the budget for idea development. That, of course, was the problem. It has been the problem ever since. This failure on the part of my management to fight for research in ideas was one of the reasons I left the Bureau.

U.S. Naval Research Laboratory, 1952-1954

At the Naval Research Laboratory, I became part of an **interdisciplinary team** applying the latest **operations-research techniques** to the development of **man-machine systems**.

Brave words.

The “disciplines” included physics, mathematics, philosophy, sociology, various brands of psychology, naval science (represented by officers in the United States Navy), and I don’t know what all.

Our tasks were the obvious ones: develop ways to make naval operations more efficient by incorporating new gadgets into systems that used them to best advantage. A typical assignment might be either general (for example, design of a combat information center for a ship, a task force, or a shore-based control center) or specific (for example, taking a new weapon, like a homing torpedo to be fired from the deck of a ship into the water near a distant submarine, and deciding how best to incorporate it into a system including a ship, sensing devices, communicating devices, and other weapons).

I learned many things during my two years at the Naval Research Laboratory (among them that clinical and experimental psychologists tend to despise each other) but for our present purposes the most important things I learned were:

- It’s hard to plan effectively for a future you can’t predict.
- No numbers are frequently better than some numbers.
- Nobody knows what’s going on [Ref. 3].

Predicting the future is, of course, particularly difficult for the military since a future requiring military actions is apt to be precipitated by some catastrophic event. I attempted to develop techniques for dealing with this by first describing the three states in which the Navy might have to operate:

- The peacetime state
- The transition state from peace to war immediately after hostilities have commenced
- The wartime state

and then describing the transformations required to convert from one state to the other in the most efficient way possible.

I didn’t get very far with this, but, if I had the responsibility for contingency planning for any organization, civil or military, I would return to the three-state model I started to develop for the Navy and build on it.

And, again, I would need an efficient notation.

It was during the time I worked at the Naval Research Laboratory that I first became aware of the ease with which people can deceive themselves with meaningless figures. This is not a criticism of the Navy. It is a criticism of virtually all modern management.

You are much better off with no numbers than meaningless ones. The minute you believe numbers uncritically, that is, without understanding how they're calculated and how well they measure whatever they're supposed to measure, you will generate a breed of employee who will produce numbers and not results. Your data-processing system will then serve not to describe reality but to lie about it.

Kaiser Steel Corporation, 1954-1961

I started with Kaiser Steel as a mathematician, a new kind of employee requiring a new job description. I started in the Fontana Procedures Department of the Controller's Division. The Industrial Engineering Department (which was in the Operations Division and thus always in a kind of uneasy rivalry with the Procedures Department) had the responsibility for administering job descriptions. I learned later that the one we filed threw them into a tizzy; they felt it was another sinister move on the part of the Controller's Division to take over the work they were supposed to do.

I left Kaiser Steel, not quite seven years later, as Director of Research and Computer Planning.

In the interim, I learned about iron and steelmaking both at Kaiser and, through industry association meetings and literature, at other American and Canadian steel plants. I learned about other industries by participation in the activities of cross-industry associations. By the time I left Kaiser Steel, I had had several years in which to observe management in action. The comments I've been making, which you may have regarded as flippancies, are honest descriptions of what I observed.

Consider some examples. They are from Kaiser Steel, because that is where I worked, but I can testify that they are representative of all management.

Precision rounds. At the time I was there, Kaiser Steel manufactured an alloy steel product called "precision" rounds because the diameters had to be controlled to very close tolerances. There were two schools of thought about the place of precision rounds in our product line. One school held that it was the most profitable product we made. The other said we were losing our shirts on it.

How could this be? Couldn't our cost accountants tell us whether we were making money or losing money?

The answer is: no, they couldn't. Two different groups, working off the same set of figures, reached diametrically opposite conclusions. This was the first of many experiences (including attendance at a Stanford gathering of the most prestigious accounting firms in the world) that led me to the conclusion that most cost accounting is applied metaphysics of an extremely ethereal kind. Counting angels on the head of a pin is a useful exercise in data-gathering compared to much cost accounting.

The problem was, as it always is, the basic data with which we had to work. Rounds of any kind had to go through a finishing operation. Scheduling finishing operations

and describing what took place was one of the most difficult tasks in the mill. Production figures out of the finishing operation were always suspect. The difference between the two schools derived primarily from the different ways they interpreted those figures. At least, that's the best explanation I was ever given.

Tin mill flippers. We installed a management incentive plan at Kaiser Steel while I was there. The incentive plan for the tin mill was delayed for a while until we straightened out a rather embarrassing problem that cast doubts on the accuracy of our tin mill production recording. We were reporting more tin plate coming out of our shears than we put into them. Management was understandably hesitant about paying performance incentives on figures that were so obviously, stupidly wrong.

The difficulty arose from the way we processed rejects. As the sheared plate went by an inspection station, it was examined for pinholes, surface blemishes and other defects. When a defect was detected, the inspector pressed a button that switched the faulty plate to a reject pile. Unfortunately, a few plates before and after the bad one were also diverted to the reject pile. These were usually prime plates that we could not afford to sell at secondary prices.

The prime plates in the reject piles were separated from the true rejects by a group of women called "tin mill flippers". They worked at large tables, examining the plated surfaces carefully and separating the plates into prime and secondary piles. The difficulty arose here. Because of the way reporting was done, some of the plates were counted more than once.

We caught this bad data because it was so obvious. A shear can't produce tin plate; it can only cut it. To shear more tin than you were given was clearly impossible.

But, of course, this kind of mistake indicated just the tip of the iceberg. The errors that weren't so obvious weren't caught and corrected. And they existed, let me assure you of that. Even worse, they were almost certainly manipulated by people who were better at doctoring figures than at making steel.

Slab inventories. Steel ingots are broken down into **blooms** if they are to be processed into products like H-beams, I-beams and the like and into **slabs** if they are to be processed into hot- or cold-reduced product like sheet and strip. Our biggest semi-finished inventory at Kaiser Steel was in slabs; we had between 80,000 and 100,000 tons scattered in piles over wide areas of the mill grounds.

Slabs are big, heavy chunks of steel. You'd think it would be hard to lose one. It's not. It happens every day. Although I've been out of the steel industry for more than twenty years, years in which we've landed men on the moon, I'm willing to bet that right now, somewhere in the world, a rolling mill is idle because it's waiting for a slab that's sitting on the ground not too far away.

My reason for discussing these examples is to illustrate the complexity of the activities we are trying to manage. This particular example, as it happens, also shows how quick management is to seek a technological rather than a methodological solution.

At the risk of teaching you more about steelmaking than you ever wanted to know, let me describe this attempt to solve a methodological problem by technological gadgeteering.

The steel produced by one open-hearth melt is called a **heat**. A major part of a slab's identification is its heat number. Since the process of producing slabs from ingots usually grinds impurities into some of their surfaces, the slabs making up a heat had to be distributed among areas called scarfing bays where men with oxyacetylene torches would burn out the impurities. The slabs then had to be reassembled into a heat; the heat had to be deposited somewhere in the slab yard; and the heat number and slab-yard location had to be reported to the production schedulers.

The existence of surface impurities that required scarfing thus caused much of the delay and confusion that attended the progression of the heat from the slab mill to the reduction mill.

The technological solution proposed for this problem was called a "hot scarfer". It burnt off **all** the surface of a slab immediately after it was reduced to its final dimensions. This was supposed to eliminate the need for splitting a heat up into scarfing bays.

Thus, at this point, management had a choice:

1. **Methodological.** Put in the time effort and money it would take to develop a satisfactory, realistic scheduling and inventory control system for slabs. This would necessarily require investigation of **all** the many sources of difficulty, not just the kind of difficulty that the hot scarfer was supposed to eliminate.
2. **Technological.** Buy a hot scarfer and hope the slab scheduling and inventory problem would go away.

No contest. The lure of the tangible, glamorous gadget always wins out over the intangible, colorless idea. They got a hot scarfer. I say "they" because this was done after I left; I am happy to say that I had no part in the decision.

To the best of my knowledge, the hot scarfer never worked satisfactorily. It is one item of evidence justifying my belief that technological quick-fixes seldom achieve their objectives. (This, incidentally, is even more true for data-processing than it is for steelmaking.) In this particular instance, the gadget was expensive to purchase and to operate, it burnt off good steel as well as surface impurities, and did not do what it was purchased to do: reduce wait-for-steel time in the mills that used slabs as inputs.

What does all this steelmaking jargon have to do with APL?

In all of my computer experience, I have found that the chief obstacle to getting anything done is the absence of any clear, concise, precise, formally manageable way to describe and analyze what we're actually doing and to describe and design a transformation to what we should be doing.

I gave several talks on this topic at professional meetings of various kinds. (I later used the written form of these talks, and other papers I wrote during my employment at Kaiser Steel, as class notes in the Business Information Systems classes I taught at the Stanford Graduate School of Business.) One of the earliest of the talks, "Formalizing Business Problems", was given at the first Electronic Business Systems Conference held

in Los Angeles in 1955. This aroused the interest of Murray Lesser, at what was then the newly established IBM facility scattered throughout several locations in downtown San Jose (the one that developed into the General Products Division in which I now work). We met to discuss what we could do to develop some of the ideas we had in common. The result of our discussions was a joint venture, called the Business Language Research Project, in which employees of IBM, Kaiser Steel and Touche, Niven, Bailey and Smart participated. My contribution to the project was something I called "field-and-branch identification" which I later developed into the approach to systematic systems analysis that I describe in my book on decision tables [Ref. 4]. I will discuss this more fully later.

Lest I leave you with the impression that nothing effective can be done about the data-processing problems of industry, let me assure you that this is not the case. Among the accomplishments of which I'm proudest are some of the systems I installed at Kaiser Steel. At least one of them, a tin-mill in-process inventory and production recording system impressed the phone company so much that, since we were using some of their equipment to record production and to communicate between processing points, they ran an ad in *The Wall Street Journal* featuring a picture of our tin mill and describing our system as an example of what could be achieved if you called in their Production Recording Consultant. We never had the benefit of a Production Recording Consultant's services because the phone company never told us they had one. Maybe he was made available to some of the people who read the ad.

So things **can** get done. But getting them done is slower, more difficult and more costly than it has to be. That is why we have the application programming backlogs that we do. We need a better, more systematic way of dealing with complex digital procedures.

We need systematic systems analysis.

Now do you see the connection with APL?

IBM, 1961-

I started to work for IBM in the Advanced Systems Development Division, San Jose. I have since worked in the Palo Alto Scientific Center, the Palo Alto Systems Center and the Santa Teresa Laboratory, where I am now in APL Development.

At the time I started my IBM employment, the ASDD library used to circulate a daily list of its acquisitions to all employees. I am a kind of pack rat when it comes to written material and I acquired all kinds of library offerings that, I'm sorry to report, I never took time to read.

Among the publications that I thus acquired, scanned, and filed for future reference were several reports containing a strange, exotic notation. I was skeptical about the value of these reports since, by that time, I not only had my own ideas as to what was needed but I had also seen many attempts by many people to develop notations, charting techniques, and other descriptive schemes. I didn't think much of any of them. By and large, history seems to have agreed with me; most of them are happily forgotten.

However, when I learned that the author of several of the papers full of Greek letters, slashes, little circles, curlicues, and other cabalistic symbols was coming out to San Jose to talk about his ideas, I decided that I'd read one or two of his papers as a preparation for his talk.

My life hasn't been the same since.

My first acquaintance with the notation that has since become APL (for several years it was either "Iverson Language", "Iverson Notation", or just plain "Iverson") started with an IBM Research Report by Kenneth Iverson called, *The Description of Finite Sequential Processes*.

I don't have the paper handy at the moment so what I'm about to tell you is all memory; it may be mistaken in details but not in essence. I seem to remember that the first page was mostly given over to heading material, possibly an abstract, so that there were only two short columns of reading matter on it. And, again memory, it took me several hours to understand what those two short columns were all about.

The author's approach was so different from anything I'd ever encountered that I had a difficult time adjusting to his frame of reference. At the end of the first page, a fair assessment of my state of mind would be that I had glimmerings but no hope.

The second page took about as much reading time as the first but, since it had twice as much matter, I was clearly improving. The glimmerings were now fitful gleams. One thing had definitely changed, however. I had no doubts about the value of what I was reading. I was now virtually certain that the author had something to say and that I'd better find out what it was.

The third page had an illustration that, in a few short lines, described George Dantzig's simplex algorithm simply and precisely.

That was the overwhelming, crucial experience.

In the previous thirteen years, I had participated in so many murky discussions of what was here presented with crystal clarity that I knew that what I was reading was of enormous significance to the future of computing.

So, when Dr. Kenneth Iverson came out to talk to us at San Jose, I was not only a convert, I had a fair idea of what he had to say. In the upshot, this meant that I was the only one who could ask him questions. Ken had some good, sharp people from Research and Advanced Systems Development in his audience but I'm pretty sure I was the only one who had been lucky enough to read what he had to say beforehand so that I had a fighting chance to follow him when he did what he usually does: hit you with one idea after another so fast that your mind goes numb.

I had been alerted to the fact that Ken might know what he was talking about by a fellow employee named Don Fisher who was working in the same group that I was. After Ken's talk, he came to visit Don and I got a chance to meet him. It's hard to believe that that first meeting took place more than twenty years ago. But it's true. I've been an APL bigot for a long, long time—since before there was an APL, in fact.

Stanford, 1962-1967

Shortly after I went to work for IBM, Dan Teichroew, a friend of mine from the Bureau of Standards days, asked if I would be interested in spending part of my time at Stanford, participating in a study of "Quantitative Management Techniques" being conducted at the Graduate School of Business. Dan was a Professor of Management

at the GSB and he wanted my permission to approach IBM about the idea. Naturally I was delighted by his proposal and even more delighted when my management gave its approval.

The next several years were among the most satisfying and productive I've ever spent. And, in my opinion, not only did I benefit from them but so did everybody else who was involved: Stanford, IBM, the students who participated in the research program and attended my Business Information Systems classes and, more to the point of this talk, APL itself, whose first implementation was a FORTRAN-based batch interpreter that was developed on the IBM 7090 in Pine Hall at Stanford.

In the *Formalizing Business Problems* talk that I had given in 1955, I asserted that the problems we were facing required a partnership among computer users, computer manufacturers and academic institutions if we were ever to develop the body of knowledge we needed to manage computers properly. For a while there at Stanford we had two-thirds of what I'd recommended and, as you will see, I concentrated a good deal of effort on seeing that the third was represented as well.

When I started at Stanford as an Industrial Research Fellow, the present School of Business building did not exist. I occupied an office in Polya Hall, near the temporary buildings used to house Business School faculty and staff. IBM, at that time, shared the 7090 with Stanford and thus had the use of a few offices in Polya Hall, the building which housed the university's Computer Science faculty and staff. I was assigned one of those offices.

This was ideal. I not only participated in the activities of the Graduate Business School; I was also part of the Computer Science complex at the university. Both of these associations had APL implications and I'd like to tell you about them.

My activities at the School of Business can be described in four parts:

1. Work with graduate students on the "Quantitative Techniques" project.
2. Guest lecturing in Business Information Systems courses taught by Dan Teichroew and John Lubin.
3. Development of my own Business Information Systems course after Teichroew and Lubin left the university.
4. Teaching Operations Research courses as a Lecturer in Operations and Systems Analysis.

APL and quantitative techniques. My purpose in describing my pre-APL history to you was to let you know how my ideas about what management needed were formed. If you were paying attention, it should come as no surprise to you that, as soon as I was given an opportunity to do something about it, I started to investigate the implications for management of an efficient notation for describing, analyzing and designing digital procedures.

I investigated two techniques: decision tables and APL. The latter is the one relevant to this talk. I've long regretted that I never wrote up what I did; let me remedy the deficiency now.

A Programming Language was published just in time for me to use in my researches at Stanford. It was a godsend. I used it to try to answer the question:

Is it possible to write programming specifications in such a way that ambiguities, misunderstandings and outright mistakes in programming are minimized?

The answer I got surprised even me.

Here's what I did. For a set of problems, of increasing difficulty, I would write a solution procedure, using the notation of *A Programming Language*. I would then ask the graduate student assigned to me to:

- program the procedure in any programming language he chose
- execute the procedure for a representative set of data values
- give me his program and answers so that I could compare them with the specifications I had written and the answers I had previously calculated

The result, which I still find surprising and impressive, was:

In every case, what was programmed was exactly what I specified.

I'll comment in a moment on how rare this is under any circumstances. But these **particular** circumstances were so extreme that they merit some discussion.

A Programming Language was and is crammed full of ideas. I studied it assiduously, and enjoyed it, but it was not easy reading, at least not for someone of my mental capacity. I was lucky that my Stanford assignment provided me some time to spend on it so that I could both use it and explain it when my students asked questions.

But think of **them**. The life of a graduate student is a busy one. They had less time than I had to study strange notations. How could they make sense of the chicken scratches I said were their programming specifications?

Somehow they did.

I gave them completely abstract procedure descriptions, even avoiding standard words where they might have provided clues to the nature of the procedure, and they programmed exactly what I specified **even though they had no prior experience with the notation and never became expert at using it.**

Thinking back on it, I understand why an intimate familiarity with this strange notation would have been desirable but was not essential. The only part of the notation they needed to understand was the part I used. I told them to look up the meaning of the symbols in Ken's book but I also told them that I'd explain and illustrate the meanings myself if they wanted help. But I wouldn't do anything but explain the symbols. They had to translate the symbolic description into a program.

They did, with no arguments and no discussions about what the procedure was supposed to do. The procedure was described completely abstractly. They had no idea what external significance it had. They were not led astray by ideas of their own about it. They figured out what I wanted done and then did it.

Contrast this with the usual way in which a business procedure gets programmed. Somebody, usually called a **system analyst**, casts about, sums up with some general ideas, puts them down as programming specifications and presents them to somebody else, usually called a **programmer**, who:

- asks the analyst lots of questions
- programs what he thinks the analyst wants
- is told that what he has done is completely wrong
- quarrels with the analyst about what he said and what it implied
- forces a reworking of the specifications
- tries another program
- is wrong again, forcing another rework of the specifications

and so on until finally, after a long period of false starts and reruns, **some** program is accepted as an implementation of **some** set of specifications, both programmer and analyst now being so sick of the entire procedure that they no longer care whether what is finally programmed is what was originally wanted.

Is what I'm describing familiar to you? Does it, perchance, occur in your organization?

To me, specifications cannot properly be called specifications until they are as abstract as blueprints or mathematical formulas. Specifications using the words of everyday speech are always subject to misinterpretation. Most of the costs, delays and other inefficiencies that attend the development of data processing procedures are due to these misinterpretations.

The abstract APL program specifications I tested at Stanford strengthened my belief in the opinion I've just expressed.

Content. What did I ask the students to program? Let me select four examples.

Rings-o-seven. The first example was from *A Programming Language*, page 63, Exercise 1.5. It was a solution of a "ring-o-seven" puzzle, in which rings on a bar were to be removed according to certain rules. The bar was represented by a logical vector in which the presence of a ring was indicated by a 1 and the absence by a 0.

This was just a warmup experiment, but it had an informative result. My programming specifications, naturally, described the solution procedure I had devised. It was wrong. **But it was programmed exactly the way I specified it.**

Think of it. No arguments about who misunderstood whom. The systems analyst was wrong; the programmer was right.

Wouldn't you like to be able to make that determination without bickering, recriminations or tears?

(After my blushes subsided, I wrote a correct procedure. It was programmed correctly and gave the right answers. I tell you this because I'm vain and don't want you to remember me for the only mistake I ever made in my whole life.)

Internal rates of return on investment. Financial theory at that time was troubled by the fact that the then current procedures for calculating rates of return on an investment gave ambiguous answers in some cases. The difficulty arose when the cash flows that characterized the investment were a mixture of positive and negative amounts. This led to an equation with multiple roots, so that, in many cases, two equally ridiculous rates of return were calculated.

Dan Teichroew felt that the difficulty lay in the assumption that the same interest rate should be applied to the negative cash flows as to the positive. What he suggested was that there would be a unique, meaningful solution if we assumed that the rate to be applied to the negative flows, which were, after all, borrowings, should be a putative “cost of money” which would **not**, in general, be the same as the return on the investment.

Given this assumption, I provided a proof that an “internal rate of return” on the investment would be uniquely determined when a “cost of money” was assumed.

I then specified, in “Iverson Language”, a procedure that determined a rate of return for a fixed cost of money and a specified series of cash flows. The actual calculation was programmed by J.P. Seagle, a graduate student. I no longer remember what programming language Pete used, possibly a home-grown (Stanford) assembler for the 1401. The results were reported in a couple of papers by authors the very ponderosity of whose names (Teichroew, Robichek, Montalbano) testified to the validity, excellence and importance of the research they described.

Critical-path calculations. Critical-path and PERT calculations were all the rage at that time, with many papers being devoted to efficient schemes for “topological sorting” and for detecting inconsistencies in precedence relationships.

I became interested in the problem and decided that the need for topological sorting could be eliminated and that consistency checking did not require a separate, special program. I used APL (Iverson language) to describe a solution procedure in which topological sorting was not required and consistency-checking was a fallout from the basic critical-path calculation.

This time, in addition to having Pete Seagle program the calculation, I programmed it myself, in MAP, the IBM 7090 assembly language, and FORTRAN (for input-output subroutines). My program exploited almost every bit in the 36-bit 7090 word. This permitted me to store enormous networks internally, so that I was able to achieve calculation speeds far in excess of any other method then available.

I was not then, and have not since become, an expert assembly-language programmer. I had risen to too august an eminence at Kaiser Steel to do much programming, though I snuck some in now and then, when no one was looking. My critical-path algorithm was the first programming I had ever done for the 7090 which, like MAP and FORTRAN, was completely new to me.

With the precise specifications of the APL procedure as my guide, programming assembly language for a machine with which I had little experience went very quickly and with no errors other than mistypings. The program I developed was a useful one that I later used in classes in the Business School and in the International Center for the Advancement of Management Education at Stanford. I described it in a paper, “High-Speed Calculation of the Critical Paths of Large Networks”, that appeared both as a Palo Alto Scientific Center report and as an *IBM Systems Journal* article. The algorithm presented in the paper used the old notation (the one in the book) since the new (the one for the typewriter) had not yet been designed.

So APL turned out to be a particularly useful way to specify a program for an inexperienced programmer—me.

Linear programming. The last program specification I want to discuss was done by a student, Don Foster, who had even less time than the general run of student I'd been working with. I believe it was his last term at the School of Business. This is always a hectic time but in Don's case he had the added distraction of planning for a European trip. Even without my assignment he was leading a harried life.

I gave him a version of the simplex algorithm (basically the one that had originally sold me on APL) from which I'd removed all clues like "unbounded", "infeasible", etc.

The solution was programmed in jig time, since Don was champing at the bit anyway. The programming language was FORTRAN. The program ran the first time it was tried. It produced the right answers.

What was interesting was Don's reaction when I told him what he'd programmed. Like all good business school students of that era, he'd received instruction in linear programming. But he hadn't been told how easy it was. The arm-waving and chalk dust had concealed the basic simplicity from him as they had from me.

Business Information Systems

As a guest lecturer in Business School courses, I advanced the argument that I've been advancing throughout this talk, that we need an efficient notation for describing procedures. I illustrated some of what could be done with APL and decision tables.

In my Business Information Systems course, I did the same, but I also encouraged activities that would permit students to find out for themselves whether or not I had valid reasons for what I was recommending. One of the course requirements was completion of an approved project. As an example of the kind of project I had in mind, I would suggest that they go to a local company, get a copy of an important report used by several departments, and visit each of the departments, asking whoever got the report what he thought the report told him, how much he knew about how the figures in the report were determined, what actions he based on the report, and how he decided on his actions.

Few of the students had had detailed business experience at that point in their careers. Business to them was defined by the other business school courses they'd taken: finance, marketing, micro- and macro-economics, accounting, theory of the firm, organizational behavior, and so on. These were good courses but none of them were concerned with, or had the time to devote to, determining what was happening at the working or first-line management level of an organization.

The intent of my course was to forewarn my students that business was, in practice, a good deal more disorganized than they were likely to realize from most academic discussions.

I had some skeptics in my classes, people who felt I was overstating my case. None of the skeptics who attempted the kind of project I suggested remained skeptics. Some, shattered by their experiences, felt even more strongly than I that no one in management knew what was going on.

Several students caught the APL bug. They went out on missionary activities of various kinds after graduation. The effects of some of these are still being felt—in, for example, organizations like IBM and American Airlines, to name the two I know most about.

Stanford's Computer Science Department

Although I was housed with members of the Computer Science Department, I had no official connection with it. All of my interaction with faculty, staff or students was informal.

From the standpoint of APL as it now is, however, this interaction was the important one. This was not because of anything I did. It was primarily because I was a reminder of the existence of "Iverson language" and a kind of catalyst who served to bring together the right people at the right time.

The Computing Science Department of those days was an ALGOL stronghold. It had a Burroughs B5000, later upgraded to a B5500, an IBM 7090, and a PDP-1, probably the first computer I ever saw with a cathode-ray tube terminal. I don't know whether "Star Wars" (the game) was developed at Stanford. I do know that a lot of "Star Wars" was played there.

Stanford had developed its own version of ALGOL, called SUBALGOL, for the Burroughs computer that had preceded the B5000. I believe the number was B220, but my memory might be playing me tricks. At Kaiser Steel, our first computer had been a B205, predecessor of the B220, if that's what it was.

The significance of this information from the APL standpoint is that two, possibly three, of the people who played key roles in developing the very first APL system had been instrumental in producing SUBALGOL for Stanford: Larry Breed, Roger Moore, and (the one I'm not sure about) Phil Abrams.

I met Larry as a result of a talk I'd given as one of a series on "Programming Languages" conducted by the Computer Science Department. He expressed an interest in what I'd had to say about what I was doing in the School of Business with the notation described in *A Programming Language*. He and Phil Abrams took action on this interest in a very real, very productive way when the *IBM Systems Journal* article, "A Formal Description of System/360" appeared. What they did, and its aftermath, is described in Appendix A, an annotated verse history of APL's early days.

Larry and Phil not only developed the batch APL interpreter I mention in the verse, they did so many other things that I wish they and others involved in APL's origins would get them down on paper. For example, one of them should tell the story of **Elsie** (for low cost), an APL mini before there were minis.

But, in essence, all I did was happen to be around, saying the right things to the right people. Things took off when the right people got together.

Incidentally, one of the people involved in the "Programming Languages" seminar to which I referred above was Niklaus Wirth. Unfortunately, Klaus didn't get the proper message from my talk. He went his own way and developed PASCAL.

APL at IBM

The history of APL at IBM has been a curious one. In the early days, those of us who believed in APL were regarded as being a little (perhaps more than a little) strange. Since much of the strangeness was concentrated in IBM Research, this was tolerated. Practical people (the kind of people who make sales and meet payrolls)

expect research people to be strange and are usually disappointed when they're not. So the strange people in Research were written off as overhead and left to amuse themselves with their incomprehensible, impractical symbols.

What that particular Research group did, of course, was produce the most solid, dependable, useful time-sharing system anyone had ever seen.

I wish I could tell you what it felt like in those early days to have the use of a system that was up twenty-four hours a day, seven days a week. No one had ever known such a luxury. People who didn't bother to investigate never believed us when we told them about it.

But some people **did** investigate what the researchers had developed and started to use it to do IBM's key bread-and-butter applications. This way of doing business was so productive that it spread like wildfire. By the time the practical people found out what had happened, APL was so important a part of how IBM ran its business that it could not possibly be uprooted. The wild-eyed researchers had produced a moneymaker. No talks about product "strategies" and the evils of language proliferation prevailed against the simple fact that:

- if you worked for IBM and
- had access to an APL time-sharing service and
- had something you wanted to get done on a computer quickly and economically

then the best way to get it done was to use APL.

I wish someone who knows the details of how that came about would write about it. I can't do it. I was three thousand miles away when it took place.

APL (called VS APL for reasons beyond the ken of mortal man) is, of course, now an IBM program product. I don't know how much more practical than that you can get.

Summary: systematic systems analysis

I could go on but I see you're falling asleep. Let me end by rephrasing what is either explicit or implicit in what I've already said.

In the preface to my book on decision tables, I say

If you wish to use digital computers effectively, the first thing you should do is digitize your procedure descriptions.

As usual, this was something I realized I was doing after I'd finished writing and took time to think about what I'd written. The key idea of the book (the potential of which, incidentally, no one has as yet successfully exploited) is that procedures can be digitized in the same sense that bubble-chamber and spark-chamber pictures are digitized for analysis by a digital computer.

A decision table is a digitized procedure description; it describes a correspondence between vectors of decision values and vectors of action values.

The particular form of the digitizing is not important. Decision tables may or may not be the most effective way to get the digitizing done. The important thing is that it be done and done in a way that permits checking for consistency, redundancy, completeness.

But how is such a digitized procedure to be developed, maintained, managed, modeled, interpreted, translated, improved, extended,...?

To me, the answer is clearly APL. If we are ever to do “systems analysis” systematically, we must:

1. Digitize our procedure descriptions
2. Manage our digitized descriptions with APL

We must divert our research to developing ideas rather than gadgets. Good ideas are at hand. Let's develop them.

Epilogue: twenty years after

The latest IBM version of APL is an Installed User Program called APL2. For those of us who had to bootleg our APL efforts within IBM for a long time, the announcement of APL2 is gratifying because it indicates the kind of management backing and recognition that we missed when we felt that APL was regarded as a limited tool for a small, specialized audience.

Management put its support on record in another significant way. IBM recently instituted awards for outstanding technical achievements. The first of these to anyone at the Santa Teresa Laboratory was just awarded to Jim Brown, manager of the group that developed APL2.

I haven't had a chance to use APL2 very much yet. I've been too busy writing a workspace manual for VS APL. So I can't pretend to an extensive knowledge of its details. But, recently, I used APL2 to do something that I haven't been able to do for 20 years.

Page 19 of *A Programming Language* describes a bank ledger that has three columns: the first column contains customer names, the second account numbers, the third balances. Unassigned account numbers had the entry “none” in the corresponding row of the name column.

Until APL2, no system available to me within IBM would allow me to form an array of that kind. Nor could I write, in any straightforward fashion, the four programs, producing reports from that ledger, that appear as one-liners in the book with which, twenty years ago, I started my investigation of “Quantitative Techniques in Management” at the Stanford Graduate School of Business.

With APL2, I was able to do precisely that. As a way of rounding out twenty years of APL history, I thought I'd show you what I did. It's contained in Appendix B.

Appendix A

APL Blossom Time—A History In Verse

My contribution to APL 81 was the verse that I discuss below. The most I'd expected, when I wrote it, was that Jim Brown might play it at some informal gathering. I couldn't anticipate what actually happened. A group including Jim Brown, Larry Breed, John Bunda, Diana Dloughy, Al O'Hara and Rob Skinner rehearsed their guitars and voices until they were of a truly harmonious sweetness and sang "APL Blossom Time" at the APL 81 banquet as part of the evening's entertainment. I.P. Sharp's Peter Wooster prepared overhead transparencies that made it possible for the audience to sing along. And I'm sure other people whose names I was never told contributed to what was, for me, an extremely heartwarming experience: the sound of people singing, laughing and giving every evidence of enjoying the words I'd written.

Despite its frivolity, "APL Blossom Time" is authentic history. I thought it might be useful to get the details on record by annotating each section of the verse.

APL Blossom Time

A nostalgic reminiscence of the early days of APL, remembered to the tune of "The Battle of New Orleans".

Back in the old days, in 1962
A feller named Ken Iverson decided what to do.
He gathered all the papers he'd been writing fer a spell
And he put them in a little book and called it APL.

Well...

He got him a jot and he got him a ravel
And he revved his compression up as high as she could go
And he did some reduction and he did some expansion
And he sheltered all his numbers with a ceiling and a flo'.

If you've read the earlier part of this book, this verse doesn't need annotating. If you haven't, go back and read it.

Now Sussenguth and Falkoff, they thought it would be fine
To use the new notation to describe the product line.
They got with Dr. Iverson and went behind the scenes
And wrote a clear description of a batch of new machines.

Well...

They wrote down dots and they wrote down squiggles
And they wrote down symbols that they didn't even know.
And they wrote down questions when they didn't know the answers
And they made the *Systems Journal* in nineteen sixty-fo'.

Though the scan required that I place Sussenguth's name first, this is perhaps misleading. Ed Sussenguth has done a lot of good work for IBM but, except for his participation in this paper, I don't know of any other in which he used APL.

Adin Falkoff, on the other hand, was one of those crazy-symbol Iverson-Language authors whose papers I started requesting from the ASDD Library when I first joined IBM. I remember one paper in particular. It struck me because it seemed to be written by someone who hated jargon as much as I did. One of the data-processing fads current at that time was the "associative memory". Adin, like the rest of us, had to use the term because everybody else was using it. But he rather wistfully (can you imagine Adin wistful?) pointed out that a more descriptive term would be "content-addressable memory".

And, of course, as you all know (or should know, if you don't), Adin Falkoff, in both technical and administrative capacities, has been in the forefront of APL developments ever since the days of those early, incomprehensible reports out of IBM Research.

The paper referred to in the verse is "A Formal Description of System/360", by Falkoff, A.D., Iverson, K.E., Sussenguth, E.H. *IBM Systems Journal*, Vol. 4, No. 4, October, 1964.

About "questions where they didn't know the answers": the paper was indeed, to the best of my recollection, the first to use the question mark as an APL function.

I gave my copy of that issue of the *Systems Journal* to Larry Breed. I had already ordered several more. John Lawrence, editor of the *Systems Journal* at the time, had the APL functions in the article printed separately for more effective study. I ordered several copies of those, as well. Larry and Phil Abrams conducted a seminar on the System/360 paper that extended over several weeks. They also produced a list of "cliches", to assist in understanding regularly recurring patterns, and a list of errata, to remind the authors (or the typesetters) that they didn't know it all.

The sessions conducted by Larry and Phil were well attended. When Ken Iverson came out to give a talk at Stanford, he drew the biggest crowd the Computer Science auditorium had seen up to that time. I told my Business Information Systems class to attend since they would hear something better than anything I had to say; this also gave me a chance to attend myself.

Now writing dots and squiggles is a mighty pleasant task
But it doesn't answer questions that a lot of people ask.
Ken needed an interpreter for folks who couldn't read
So he hiked to Californ-i-a to talk to Larry Breed.

Oh, he got Larry Breed and he got Phil Abrams
And they started coding FORTRAN just as fast as they could go
And they punched up cards and ran them through the reader
In Stanford, Palo Alto, on the seventy ninety-oh.

Ken Iverson and Larry Breed first met in my office at Polya Hall. Since this may be my only claim to fame, I'm glad to put this historical fact on the record.

Larry was about to graduate. Ken had a job to offer him. We now have APL.

I remember a phone call of Ken's, shortly after Larry had joined him and Adin at IBM Reseach in Yorktown, in which he said something like: "This young man thinks he can write a translator in a couple of months." He sounded as if he were wondering whether he'd made a bad bargain. I assured him that if Larry said he could do something in a couple of months he would probably do it in a couple of weeks. He and Roger Moore were legends in their own time during Stanford's SUBALGOL days.

Stanford has left a mark on APL second only to that of left-handed Canadians. For a while, there was a theory that all of APL was being dominated by left-handed Canadians. I have been told that when Mike Jenkins, at lunch in the Yorktown cafeteria, was observed to be left-handed, someone facetiously asked him if he happened to be Canadian. He happened.

I tried to start a similar factoid [Ref. 5] about right-handed Brooklynites, hoping to get included along with Falkoff and McDonnell. I forget what happened to that. I think one of them is left-handed. I know I'm not.

Well a FORTRAN batch interpreter's a mighty awesome thing
But while it hums a pretty tune it doesn't really sing.
The thing that we all had to have to make our lives sublime
Was an interactive program that would let us share the time.

Oh, they got Roger Moore and they got Dick Lathwell,
And they got Gene McDonnell with his carets and his sticks,
And you should've heard the uproar in the Hudson River valley
When they saved the first *CLEANS*PACE in 1966.

APL bigots seem to be characterized by literacy and a feeling for history. The first time-sharing APL system was implemented (as IVSYS) on an IBM 7090 at Mohansic. In those days, there was no *)CLEAR* command. To get a *CLEAR* workspace, you had to load one. The one that came with the system was called *CLEANS*PACE. Although it was no longer needed when *)CLEAR* was introduced, *CLEANS*PACE, along with the time and date it was originally stored, has been preserved in Library 1 of a continuous sequence of systems ever since: the Yorktown Model 50, the Philadelphia Scientific Center Model 75, Palo Alto Model 158, the Santa Teresa Model 168 and, as I discovered for the first time just a few hours before I wrote this, the Santa Teresa

Model 3033. At one point, after a disaster had caused the loss of *CLEANSPLACE*, it was carefully restored with the correct date and time. The objective, of course, is to preserve a record of the moment when APL first became a time-sharing computer language.

Preserving *CLEANSPLACE* in APL2 presented a problem, since workspace names are limited to eight characters in CMS, the first “environment” in which APL2 has been offered. However, as you can see from the following exhibit, which is a copy of what appeared on the screen in response to a *)LOAD 1 CLEANSPLACE* command that I executed on our APL2 system (which operates under CMS) the problem has been solved, or, better, circumvented.

```
)LOAD 1 CLEANSPLACE
SAVED 17:53:59 11/27/66 (GMT-5) 1351K(32K)

  □TZ
-7

)COPY 1 CLEANSPLACE
SAVED 17:53:59 11/27/66 (GMT-5)
```

Note that not only is the time given, the time zone of the area in which the storing was done is also given, indicating that the original workspace was stored when United States Eastern Standard Time was in effect. Note also that the original workspace size was 32K and that the time zone in which *CLEANSPLACE* was loaded to produce this example was Santa Teresa Daylight Savings Time.

What hard workers APL bigots are! I’ve checked my handy perpetual calendar and, as far as I can tell, November 27, 1966 was the Sunday of what must have been a four-day Thanksgiving weekend. What were those loonies doing working such crazy hours during the holiday season?

The “carets and sticks” reference is to a paper by Gene McDonnell on the logical and relational functions—the ones whose symbols can be constructed out of “carets and sticks”.

Well, when Al Rose saw this he took a little ride
In a big station wagon with a type ball by his side.
He did a lot of teaching and he had a lot of fun
With an old, bent, beat-up 2741.

Oh, it typed out stars and it typed out circles
And it twisted and it wiggled just like a living thing.
Al fed it a tape when he couldn’t get a phone line
And it purred like a tiger with its trainer in the ring.

Al Rose was, and I assume still is, one of the most spectacular APL demonstrators there ever has been. The verse refers to a vacation he took in which he was accompanied not only by his family but by what was laughingly called a portable 2741. This was a 2741 that came in two parts which, when ready to be “ported”, looked like two big, black pieces of luggage. Wherever Al went, he’d find some likely APL prospects, park the station wagon near an electrical outlet and a phone, lower the tailgate and start hammering on the keys.

In those days, getting connected to a working APL system was a chancy thing. As a safeguard, Al recorded, on tape, what went across the acoustic coupler during a sample session. When he had problems getting to a real APL system, he'd play the tape into the acoustic coupler and produce a simulated computer session that was an exact copy of the real thing.

I remember that double-black-box 2741 very well myself. I, too, did quite a bit of APL demonstrating in those days. At the University of California at Davis, the demonstration was given on the second floor and there was no elevator. I had to haul those two big boxes up a long flight of stairs. I'm glad I didn't find out until later how heavy they were. When I sent them Air Express to an IBM System Engineer in Seattle, I learned for the first time that they weighed 120 pounds. Well, I'm not too bright but I'm pretty strong.

Now, there's much more to the story, but I just don't have the time
(And I doubt you have the patience) for an even longer rhyme.
So I'm ending this first chapter of the tale I hope to tell
Of how Iverson's notation blossomed into APL.

So...

Keep writing **nands** when you're not writing **neithers**,
And point with an arrow to the place you want to be,
But don't forget to bless those early APL sources
Who preserved the little seedling that became an APL tree.

Dedicated to the pioneers of APL with respect and
affection by

J.C.L. Guest

J.C.L. Guest is the pseudonym I used for some light verse I submitted to *Datamation* several years back. There were four pieces in all: *The Far-flung Data Base*, *SYSABEND Dump*, *Virtual Memory* and *Decision Making*.

If you were offended by the unkind things I said about modern management in this talk, don't read *Decision Making*. You won't like it.

Appendix B

Twenty Years After

The following four figures show how I applied APL2 to Program 1.9 (Example 1.1) of *A Programming Language*, page 19.

The first figure shows the sample bank ledger I used and the calculations I performed to illustrate the ledger's shape and various facts about its composition. Note that the name entry for an unassigned account number is a single blank rather than the "none" used in the original example.

The second figure shows two versions of the four reports (P, Q, R, S) required in the example. In the first, the output is unformatted. In the second, "picture format" is used to format the numeric part of the report.

The four required reports are:

1. P —name, account number and balance for each account with a balance less than two dollars. (Although the original example did not require this, the illustrated calculations do not include unassigned account numbers in the report.)
2. Q —name, account number and balance for each account with a negative balance exceeding one hundred dollars.
3. R —name and account of each account with a balance exceeding one thousand dollars.
4. S —all unassigned account numbers.

The third and fourth figures show the programs for the unformatted and formatted reports respectively. They could have been written as the four one-liners of the original example, except that report P , the one producing a list of accounts with balances of less than two dollars would have included unassigned account numbers. To avoid this, the unassigned account number report, S , was produced first and an array T , consisting of all assigned accounts, was used to create the subsequent reports.

The other lines in the report merely introduce spaces to separate the successive reports.

Figure 1

Bank Ledger Based on Example on Page 19 of A Programming Language

```

    LEDGER
PAUL DOMBEY      1  125912.67
WILKINS MICAWBER 2  -3333.33
MORLEENA KENWIGS 3    1.97
RALPH NICKLEBY   4  65555.32
                  5    0
MARTIN CHUZZLEWIT 6   426.5
NICHOLAS NICKLEBY 7    1.02
JONAS CHUZZLEWIT 8   3333.33

    ρLEDGER
8 3

    LEDGER[;1]
PAUL DOMBEY WILKINS MICAWBER MORLEENA KENWIGS
RALPH NICKLEBY MARTIN CHUZZLEWIT NICHOLAS NICKLEBY
JONAS CHUZZLEWIT

    ρLEDGER[;1]
8

    LEDGER[;2]
1 2 3 4 5 6 7 8

    LEDGER[;3]
125912.67 -3333.33 1.97 65555.32 0 426.5 1.02 3333.33

    2×LEDGER[;2]
2 4 6 8 10 12 14 16

    2×LEDGER[;1]
DOMAIN ERROR
    2×LEDGER[;1]
^^
```

Figure 2

Formatted and Unformatted Calculations Described in Example on Page 19
of A Programming Language

```
PG19APL
REPORT P:  ACCOUNT BALANCES LESS THAN $2.00

WILKINS MICAWBER  2  -3333.33
MORLEENA KENWIGS  3      1.97
NICHOLAS NICKLEBY 7      1.02

REPORT Q:  ACCOUNTS OVERDRAWN MORE THAN $100.00

WILKINS MICAWBER  2  -3333.33

REPORT R:  ACCOUNTS WITH BALANCES OVER $1000.00

PAUL DOMBEY      1
RALPH NICKLEBY   4
JONAS CHUZZLEWIT 8

REPORT S:  UNASSIGNED ACCOUNT NUMBERS

5
```

```
PG19APLPF
REPORT P:  ACCOUNT BALANCES LESS THAN $2.00

WILKINS MICAWBER    0002      -3,333.33
MORLEENA KENWIGS    0003          1.97
NICHOLAS NICKLEBY   0007          1.02

REPORT Q:  ACCOUNTS OVERDRAWN MORE THAN $100.00

WILKINS MICAWBER    0002      -3,333.33

REPORT R:  ACCOUNTS WITH BALANCES OVER $1000.00

PAUL DOMBEY      1
RALPH NICKLEBY   4
JONAS CHUZZLEWIT 8

REPORT S:  UNASSIGNED ACCOUNT NUMBERS

5
```

Figure 3

Unformatted Ledger Calculation Program

```

      VPG19APL[[]]V
[ 0]  PG19APL;A S T
[ 1]  S←(T←LEDGER[;1]∈' ')÷LEDGER
[ 2]  A←(¬T)÷LEDGER
[ 3]  'REPORT P:  ACCOUNT BALANCES LESS THAN $2.00'
[ 4]  ' '
[ 5]  (A[;3]<2)÷A
[ 6]  2 1p' '
[ 7]  'REPORT Q:  ACCOUNTS OVERDRAWN MORE THAN $100.00'
[ 8]  ' '
[ 9]  (A[;3]<¬100)÷A
[10]  2 1p' '
[11]  'REPORT R:  ACCOUNTS WITH BALANCES OVER $1000.00'
[12]  ' '
[13]  (A[;3]>1000)÷A[;1,2]
[14]  2 1p' '
[15]  'REPORT S:  UNASSIGNED ACCOUNT NUMBERS'
[16]  ' '
[17]  S[;2]
```

Figure 4

Ledger Calculations Using Picture Formatting

```

VPG19APLPF[[]]V
[ 0] PG19APLPF;A S T B
[ 1] B'550555 -555,555,551.00'
[ 2] S←(T←LEDGER[;1]€' ')÷LEDGER
[ 3] A←(∼T)÷LEDGER
[ 4] 'REPORT P: ACCOUNT BALANCES LESS THAN $2.00'
[ 5] ' '
[ 6] T←A[;3]<2
[ 7] (T/A[;1]),B÷T÷A[;2 3]
[ 8] 2 1p' '
[ 9] 'REPORT Q: ACCOUNTS OVERDRAWN MORE THAN $100.00'
[10] ' '
[11] T←A[;3]<100
[12] (T/A[;1]),B÷T÷A[;2 3]
[13] 2 1p' '
[14] 'REPORT R: ACCOUNTS WITH BALANCES OVER $1000.00'
[15] ' '
[16] (A[;3]>1000)÷A[;1,2]
[17] 2 1p' '
[18] 'REPORT S: UNASSIGNED ACCOUNT NUMBERS'
[19] ' '
[20] ' ',S[;2)

```

References

1. von Neumann, John. *First draft of a report on the EDVAC*. Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, Pa., June 1945.
2. Iverson, Kenneth. *A Programming Language*. John Wiley and Sons, 1962.
3. In making this comment here, and not earlier when I was describing the work I did for the Air Force, I do not mean to make an invidious comparison of the services. The work I did for the Air Force didn't require me to know whether either the data we used or the answers we calculated corresponded to reality; all I had to do was to supply plugboards or programs. At the Naval Research Laboratory, on the other hand, it was my responsibility to get good answers from good data. That's when I found out there wasn't any good data.

To even things out, let me observe that the Air Force doesn't know what's going on either. As for the Army, well, I served in the Army. I can tell you about the Army.

4. Montalbano, Michael. *Decision Tables*. Science Research Associates, 1974.
5. Gerth, Nancy. *Adventures in Waiting*. Manuscript in preparation.

NETWORK MANAGEMENT TOOLS

Roger Moore
Vice President
I.P. Sharp Associates Limited
Toronto, Ontario

APL is often used from terminals. This raises a requirement for a method of connecting terminals to an APL system. Various methods for connecting terminals to an APL system exist. When the distances from terminal to APL system become large, some scheme for sharing of communication links becomes economically important. The requirement of shared communication links reduces the number of network technology choices. The I.P. Sharp network also has the constraint that one terminal must be able to access several APL systems. The terminals used on the I.P. Sharp network are asynchronous terminals with erratic bandwidth requirements. This combination of requirements is normally met either by packet-switching or sophisticated statistical multiplexing systems. Both technologies can meet the requirements of shared communication links and multiple hosts. Immunity from communication link errors and system overload is standard in both systems. The I.P. Sharp network uses packet-switching. Most public networks such as Datapac, Telenet, Datex-P, etc., also use packet-switching.

A packet-switched system has several types of components. Internal communication within the network is via packets. Packets are transmitted between network nodes. Communication links are the media used to transmit packets between adjacent nodes. A packet can be transmitted between two non-adjacent nodes by packet forwarding. A packet is forwarded over several communication links from the originating node to the destination node. Most traffic within the network is output from a T-task to a terminal. Two different kinds of computers are used as network nodes. The majority of the nodes are Alpha computers from Computer Automation Inc. These serve as originating nodes and have terminals connected to them. The IBM 3705 is used for connection to an APL system. Destination nodes are normally 3705s.

The network comprises 143 links and 131 nodes. With this scale of operation, a support system is required. Support is provided by 18 people in four cities and various programs and data bases. Except for the programs resident in the network nodes, all of the software to support the network is written in APL. All of the data bases are stored as APL files. Use of APL for network support has been fairly successful and has contributed to the steady growth of the network.

The data bases which describe the network and its operation can be divided into two categories. The "offline" data bases are maintained from terminals. The terminals are usually connected to the network and there may be provision for simultaneous update

from multiple terminals. The important aspect of these "offline" data bases is that they have no special connection to the network. The "online" data bases are fed by the network. Events within the network result in changes to these "online" data bases.

Program preparation and loading

The oldest network support programs deal with program preparation for network nodes. Independent systems exist for the two types of computer used in the network. This allows program preparation to be performed from any APL terminal. The Alpha software is written in a conventional line-oriented assembly language. The 3705 software is written in a medium-level language which is processed by a simple compiler rather than an assembler. Both programming systems include schemes for managing source and object programs. The final 3705 object program is eventually transferred to an MVS load library. A simple MVS program moves the object program across the channel interface to the 3705.

In the very early days of the network, several clumsy methods were used for loading programs into the Alphas. Reading a hundred feet of paper tape at 10 cps was the worst. Some of the others involving floppy disc or changing a printed circuit card were not much better. All of these methods depended on a rather cumbersome system for sending a program across a single communication link. The need for a network-oriented system such that an Alpha in Stockholm could be easily reloaded was painfully obvious. After some discussion, a method for moving the object program image through the network while sustaining normal terminal traffic, except to the node being reloaded, was specified. Some method for moving the object program from an APL file to the network was required. An N-task with a special connection to the network is used to link the APL files involved in loading to the network. The down line load task exists today as `PORT DLL` in approximately its original specification. It is responsible for converting the object program into a sequence of load packets which are sent to the node adjacent to the node being reloaded and thence to the node being reloaded. It also reformats the returned packets into a core dump. The core dump is written to file for possible analysis. Some other duties include deciding which node has requested reloading and object program customization. The exact version of the loader which is receiving the load packets is sometimes of interest. If the APL task decides that the wrong loader is being used, it loads the preferred loader and forces use of the new loader.

Down line load has been a fairly satisfactory APL system. The interface with the network is via the 370 byte multiplexor channel and a four thousand byte buffer in main store. One block of statements fills the buffer with load packets and a channel program. The channel program sends the load packets to the network and fills the buffer with dump packets. This process takes around ten seconds (exact time depends upon network delays). A few more statements extract the dump packets from the buffer and reformat the information as 16 bit words. The ability to process many object program words with a few APL statements results in a reasonable execution cost. The task is usually waiting for input/output completion. The most common state is to wait for a load request to arrive from the network. In this "wait for work" state, the loader uses about the same resources as a T-task connected to an abandoned terminal.

A minor drawback to the load scheme is that it produces a core dump of every Alpha which it reloads. These are occasionally useful for analysis of hardware or software problems. In practice most of the three megabytes per week of dumps are useless and have to be discarded to conserve file storage.

Network parameters

One problem associated with loading Alphas is that the nodes are not strictly identical. Every node has a unique number assigned to it. Two different types of communication hardware might be installed in the same Alpha. Some customization of software is required for the different types of hardware. Sundry other parameters control logging, low speed line configuration and some special features which are not present in all nodes. The original solution to customization was to link a slightly different object program for every node. APL software to describe phase customization was introduced in 1976. The following quotation from the user documentation explains the need:

The growth of the IPSA/ITS concentrator network from two nodes to more than twenty has been possible only by centralized configuration control. The satisfactory operation of the network requires that all nodes be loaded with globally consistent route tables. Convenient maintenance of the software requires that the number of custom modules and phases be kept to a minimum. Local requirements sometimes dictate special features (the American TTY problem is a good example).

To meet the twin goals of minimizing the number of phases in the system and allowing local requirements (especially route tables) to be satisfied, the solution of patching a phase during loading has been adopted for the Alpha nodes. This solution has the advantages of late binding and separation of most site dependent material from site independent material (such as the executable code). It has the drawback of being in a different format than the executable code and thus requiring specialized display and update functions. This document attempts to describe the functions which have been provided.

Some of the original network control parameters have vanished. The Teletype problem was circumvented by software modifications which have made the concentrator immune to failures in the Teletype interface. Route table calculation was a very important part of configuration control until 1981. The original routing algorithm had a strong dependence upon globally consistent route tables. As the network topology became more complex, the APL functions to compute consistent route tables became more complex. In 1981, the routing algorithm was drastically changed and the need for route tables evaporated.

The central network control file remains as a convenient repository of network parameters. About twenty people are allowed to alter it; all users have read access. For a particular node, the following items are stored:

- 1) Node name (usually geographical location)
- 2) Name of the APL file which contains the program to be loaded into the node
- 3) Destination for logging messages
- 4) Hardware used on every network communications link of this node (a binary-valued parameter)
- 5) Baud rate and hardware type for every asynchronous communication line
- 6) Destination node for optional Tally printer
- 7) Public network type for X.25 interface nodes

The above lists all the node parameters which are in use at the present time (summer 1982). These node parameters and the applicable link parameters are used by the down

line load task to customize the object program when it is loaded into an Alpha. One component in the file is an integer matrix with one row for every communication link in the network. The parameters which describe a single link are:

- 1) The node numbers for the two endpoints of the link
- 2) The line numbers within the endnodes of the link
- 3) The approximate delay time imposed by the link (normal, submarine, or satellite)
- 4) The link speed in bits per second
- 5) Theoretical worst case acknowledgement delay in milliseconds (computed from previous two parameters)
- 6) Class of service (used in alarming system but not the online network)

Adding a new node

Addition of a new node requires that the network control parameters be specified in the network control file. The parameters required by the 1 *TS* workspace are also entered at this time. A new node usually implies a new communication link. Some confirmation that the communication link is usable is desirable before attempting to proceed with the installation. The usual testing method is to connect one end of the new link to the network in its permanent location. The link termination for the new node is then placed in state called "loopback". When the link is looped back upon itself, the existing network node should receive its own transmissions. If the node detects receipt of its own transmissions an event message is sent to the logging system indicating that a particular link is in loopback. With this assurance that the link is operational, the link can be connected to the new node. It is possible to ship a node with the proper object program loaded into core storage. In this case the node will be in communication with the network shortly after it is attached to the communication link and switched on. If the machine was not shipped with the proper program, a simple console procedure can be used to initiate a reload from the APL down line load task. (If the console is defective or absent, the load can be started with a judiciously applied paper clip). The progress of the load can be monitored from the console lights. Program loading normally requires from two to five minutes. When loading is complete, the connection to the network is automatically initialized and usable for data transmission.

The node installer will usually attach a terminal to the node at this time to confirm that it does indeed support normal traffic. A node may have between four and twenty-eight terminals connected to it either directly or via dial-up modems. Each of these requires a cable from the Alpha to the terminal or modem. All of these connections have to be tested by attempting to sign-on to APL. Testing of the terminal connections may reveal some boards in the node to be faulty and replacement might be required. After all of the terminal connections have been tested, sundry "paperwork" remains. This takes the form of signing on and updating several data bases which further describe the node. None of these are used in the online network but they are rather useful in the day to day administration of the network. These administrative data bases are fairly simple and specialized. They include the following kinds of information:

- 1) Communication link repair: Most of the links in the network use circuits leased from a telephone company or PTT. The provider of the circuit has a serial number for the link which must be used when reporting a fault on the link. One data base provides a circuit number and trouble reporting phone number for every network link which terminates in a particular city.

Trouble reporting numbers are also provided for the dialup circuits which connect terminals to the node.

- 2) Replaceable parts: A typical node has about twelve field replaceable parts. The serial numbers and exact modification level of these capital goods must be recorded in a data base. Defective parts detected during installation must also be recorded in the data base. (Some of this work is often done before the node is shipped.)
- 3) Low speed documentation: The connections of terminals to the node must be documented. Every network port has a unique number which is visible to the APL user as (2 \square WS 3)[\square IO+9]. There is a data base which relates that port number to a specific telephone circuit or hardwired terminal identification. Updating this data base is part of the installation job. This data base is used for two purposes. When a fault is reported in a specific terminal or telephone line, knowledge of the associated port number is useful in problem diagnosis and repair. Statistical information about port usage is maintained and analyzed. The primary purpose is to monitor usage of dial-in facilities. If all dial-in ports in a particular city are often in use, extra ports should be ordered and installed. Similarly an unused dial-in port may indicate excess capacity (or a defective port). Both overuse and underuse are conditions which should be monitored for efficient management of the network. This requires accurate documentation of the cabling so that hardwired terminals in an I.P. Sharp branch office are not confused with the dial-in ports.
- 4) Pending installs: Installation of a new node usually implies installation of new telephone lines. A small data base lists pending installations and removals of telephone lines. The new lines are marked installed for control of telephone company invoices.

Network logging

One major problem in 1976 was ascertaining whether a particular node was operational. The desperate solution of sampling *PORTS* was used for several months. The original concentrator had some provisions for generating event messages and logging them on a Teletype connected to some node. This scheme was slightly modified by replacing the Teletype with an APL T-task. Logging messages originating in various network nodes are forwarded to the logging task. The network logging task analyzes and stores these messages. Storage is in APL files which can be read by any user. The logging messages fall into three categories:

- 1) Event messages are emitted when a node detects an event worth logging.
- 2) Statistical messages are generated at regular intervals by all nodes.
- 3) Some messages are replies to query messages emitted by the logging task.

An event message often records the failure or restoration of a network link. Event messages are normally written to file within ten seconds of the event. This is almost as fast as Teletype logging. It has the additional advantage of not being tied to a specific workstation. Any terminal can examine event messages which have been recorded in the file. Distributed access to the central event data base is quite useful. A substantial amount of fault analysis is possible simply by examining stored event messages. If all of the communication links connecting a particular node to the network are out of

service, a reasonable inference is that the node itself has failed. The ability to obtain this information from any terminal with a connection to the APL system greatly assists in repair of faults.

Statistical messages record link and network behaviour. Link measurements are made by incrementing counters. The counters are periodically sampled and zeroed. Received and transmitted packets are counted. Packet retransmissions and line errors are also counted. All of the statistics are formatted into numeric matrices and appended to a file. To avoid particularly small components, data is buffered in the workspace until about ten thousand bytes of data have been accumulated. This buffering may delay logging up to eight minutes. The logging task examines the statistical data to detect links with particularly high error rates. These links and the corresponding error rates are flagged by a special status variable.

Statistical data accumulates at over half a megabyte per day. Some of the detailed statistics are useful for investigating specific problems. Much of the data is quite boring and worthless. A daily B-task attempts to preserve the interesting data and drop the detail. Detailed data is retained for two days. Two different methods of identifying "interesting" data are used. Medium and high error rate data is collected for the entire year. Peak traffic information is also preserved. Little software for subsequent processing of the statistical data has been provided. Functions for extracting subsets from the file and displaying the matrices constitute the bulk of the support. Simple APL manipulations of the data allow the user to arrange data according to his current needs. APL expressions to find measurements or combinations of measurements which exceed a threshold are easily constructed. Selected data can be sorted or otherwise massaged with trivial APL statements.

The logging task attempts to analyze certain event messages to determine current network topology. Logging within the network uses a pyramid of nodes to reduce the number of logging calls terminated in individual nodes. The logging task is at the apex of the pyramid. The logging calls are established from bottom to top. When a call is established, status reports from all nodes in the sub-pyramid are forwarded upward. Sign-on of the logging task allows calls to be established to the highest level of the pyramid. This results in status reports from all nodes in the network. A status report lists the network links terminating in the node including the name of the adjacent node and the link status. The actual network topology can be derived from these status reports. Comparison of the observed topology with the theoretical topology from the network control file is often useful. Links which are missing in the actual topology can be assumed to be out of service. A cabling error at a particular node may have permuted the line number to adjacent node correspondence for the node. This miscabling is not particularly serious and does not interfere with normal network operation. It does interfere rather seriously with down line load as the APL program searches the link parameter matrix to determine which node is being reloaded. The search uses the line number within the node adjacent to the node being reloaded as the search argument. The error can be corrected by simply changing the network control file to reflect the actual topology.

Some network control capabilities are embedded within the logging task. These operate upon command from some task within the APL system. Examples include alteration of minor node parameters and inspection of tables within a node. All control tasks read a request from file or shared variable and report a result back to the request source. To service a request, the logging task establishes a call to the node and examines various storage locations in the node. For some types of requests, the exact locations examined in the later stages of request service may be determined by the results of

prior examinations. Requests which alter node parameters sometimes require examination of the current state of the node to refine the subsequent commands. There was also a requirement to process several requests simultaneously.

A crude multi-programming system is used to service these requests. The variables associated with a specific request are tucked away in a package when a request is awaiting input. When input arrives, a function which depends on request type is called with the current input packet for the request as an argument. The function analyzes the input and alters the variables associated with the request. The function may emit certain types of packets to the node being examined. The function may also indicate successful completion of the request to the supervisory system. At request completion the package containing the variables is returned to the request source.

Other support software

The logging task is supported by several other workspaces. There are two different workspaces for presenting link status information. The *MONITOR* workspace attempts to display current and recent status. The emphasis is on displaying conditions which might require manual action. Examples would be links or nodes which are not currently operational. A CRT is normally used for the monitor display. With a finite number of lines on the screen, conservation is desirable. An early step was to introduce a "service status" for every link in the network. Service status roughly corresponds to geographical area. The principal divisions are Europe and North America. A special status of "test" is used for certain links. Links with service status of test are links whose condition is of relatively little interest. Some of these links connect hardware or software test nodes to the network; others represent planned links which have not yet been installed. The current network contains fifteen links with test status. The network monitor never displays the status of the test links. There are provisions for further selection by service status so that display of European link status can be suppressed on North American screens. One universal need in an alarming system is some method of acknowledging alarms. Any authorized user of the monitor system can enter a line of text to provide extra information about some event on the screen. Examples include estimated time for repair, phone company reference number for the trouble, scheduled outages and various other things. The brief notes which are sometimes amplified by mailbox messages provide an adequate alarm acknowledgement system.

The other scheme for displaying link status is oriented towards hard copy reports. When a link fails the nodes at both ends of the link generate link failure messages. When the link becomes operational again, both links signal the improved status. Thus one link outage can generate four different event messages. The reporting workspace attempts to gather all messages referring to a single link in a single day and build a link by link report of incidents. The report includes failure codes and outage duration. The reason for failure may be different at the two ends of the link. Both codes are useful in problem diagnosis. The three most common codes represent: reset request from other node, timeout with loss of carrier, timeout with good carrier. A code pair such as: reset request/timeout with good carrier suggests one-way transmission difficulties. The node which received the reset request could receive properly but its transmissions were not received by the other node. A pseudo code indicates link reset due to a power failure in a node. Examination of this report provides a daily summary of network faults. Selective reporting to examine the behaviour of a particular node in a specific time period is also possible.

Sundry other support tools exist. There are workspaces for analysis of coredumps from network nodes. An attempt is made to match the observed contents of storage with the expected contents of storage. This is often useful in identifying failures in the core storage system. Certain other hardware errors with known “dump signatures” are also flagged by this workspace.

A somewhat fragile workspace attempts to draw a complete diagram of network topology. The preferred display device is an APL terminal or printer. This tends to restrict the number of angles at which links can be drawn to eight. The present result is a rather precarious tangle which bears no relation to geography. It does manage to present the detailed topology of the network in a form which some people consider usable.

Private workspaces for various special reports also exist. Many of these look at various network parameters and logs from the previous 24 hours and select features of interest to the workspace author. Examples include reloads, topology changes, high retransmission rates and various other things.

Another source of network statistics is from the APL systems rather than from network nodes and terminals operated by communication department staff. At every sign-off a record of the APL session is written to a file called the “sign-off history file”. The record includes the network port from which the session originated, sign-off time and session duration. Other information such as characters transmitted and received and billing information is also included. By merging the sign-off history records from all inhouse systems, the occupancy of network ports over a time interval can be obtained. This dynamic data when combined with the static node cabling data described above allows usage of a specific group of ports to be monitored. Another use of the merged sign-off history file is to compute traffic between an originating node and a specific APL system. This information is used for network balancing purposes.

A TOOLKIT FOR TIME SERIES OPERATIONS

Richard A. Boulay
Group Manager
Global Energy and Minerals Group
The Royal Bank of Canada
Calgary, Alberta

Introduction

A time series can be conveniently pictured as a vector with elements representing the measurement of a condition through time. The time intervals between elements need not be equal—the rhythm of seismic signals from a volcano, for instance. But most time series are regularly spaced (timed?), either by nature of their occurrence or by an arbitrarily imposed sampling frequency. As examples, growth ring variations in a giant Redwood or the daily closing value of the Dow Jones Industrial Average.

Today, I intend to focus on practical aspects of time series analysis, especially the business of extracting raw information from databases and lodging it in a form suitable for processing. The methods described here are specific to the I.P. Sharp system. However, the data management principles most likely apply to other systems as well. I will also be discussing database retrieval costs since the design and use of extraction functions should be made with some knowledge of costs to be incurred. The treatment of analytical methods will be restricted to a description of the arithmetic moving average which can be used as a filter. Finally, an annotated short list of references is included. These references are recommended as being excellent sources of information which emphasize practical applications rather than theory.

Retrieving the data

In time series work, the first step consists of extracting the information from the database. An inspection of the function coding presented here (Appendix 1) quickly reveals that I assume the user has a general knowledge of APL and is familiar with the host computer system. The code contains very few input checks or remedial provisions. It is taken for granted that the user is able to diagnose and react to error signals generated by the computer system itself. From time to time I refer to the costs generated by specific procedures. Unless otherwise specified, these refer only to CPU costs (i.e., no consideration is given to connect or character charges) determined according to the I.P. Sharp Associates rate schedule during August 1982. The rate is \$0.45/CPU for terminal tasks (T-task), \$0.35/CPU for delayed processing tasks (N-tasks), and \$0.25/CPU for batch processing tasks (B-tasks). Since all of the procedures described here

can be activated in T-task, N-task or B-task mode, their respective costs are given in all instances, e.g., \$0.70/\$0.55/0.39.

Cost Cutter No 1: In time series work there is rarely ever any justification for using a T-task. N-tasks will cut your CPU costs by 22%, and your results are only delayed by a few seconds.

Suppliers of database information invariably insist that information retrieval be accomplished using tamper-proof software which prevents the user from damaging the database. The I.P. Sharp system contains two access protocols: MAGIC and RETRIEVE. MAGIC offers an English-like command language. It is designed for users who wish to extract data and then to tabulate, manipulate and display the information by means of software resident in public workspaces. RETRIEVE provides a no-frills way of extracting data for use with user defined software. The information is returned either as a package or as raw matrices in the active workspace.

Workspace *TS1* contains our retrieval gear as well as assorted utility functions, and a few other features which will be described later. To start building your own *TS1* workspace, do the following:

```
)LOAD 55 RETRIEVE

)ERASE DESCRIBE

□LX←10

)WSID TS1

)SAVE
```

The functions *SETSTATE* (see also *SETSTATE2*), *TIMEFRAME* and *GETDATA* are user defined and serve to extract database information using the RETRIEVE protocol. (See Appendix 1)

Immediately following loading of workspace *TS1* it must be conditioned by specifying certain state settings required by RETRIEVE. Function *SETSTATE* or *SETSTATE2* (*SETSTATE* costs less) specify:

- 1) We wish to access the "NASTOCK" (North American Stock Market) database.
- 2) The data is to be returned in the form of global variables in the active workspace.
- 3) Errors are to be returned as explicit results from RETRIEVE functions, and
- 4) RETRIEVE access functions are to remain in the workspace.

Cost Cutter No. 2: When specifying the RETRIEVE state, elect to keep the access functions in your workspace. This takes up some space (about 18K) but reduces costs after the first retrieval since it cuts down on the number of file reads per retrieval.

Cost Cutter No. 3: In a dedicated production workspace it is only necessary to activate the *SETSTATE* function once. Consequently, *SETSTATE* should not be called by any other function which is to be used more than once per workspace session.

The most convenient way of conditioning a dedicated production workspace is to activate *SETSTATE* by means of the latent expression which guarantees automatic setting of the state variables upon loading the workspace. For example, a function *QUADLX* might be defined as:

```

      ∇  QUADLX;T
[1]    T←TERMINAL 'NOIDLES'  A CONDITION TERMINAL
[2]    SETSTATE A SET RETRIEVE STATE
      ∇

```

and then used as a control for the latent expression by entering the following commands:

```

□LX←'QUADLX'
)SAVE

```

Conditioning the workspace using *SETSTATE* costs \$0.81/\$0.66/\$0.45.

Having conditioned the workspace, it is now necessary to specify an access timeframe. *TIMEFRAME* is a partially conversational function which asks for date constraints in the form of logical qualifiers to the global variable *DATES*. Any legal expression will be accepted. For example:

```

DATES≥810315
DATES<810201
(DATES>811201)^DATES≤820215

```

DATES is initially defined, using a *RETRIEVE* search function, as the maximum number of daily (in this case) dates available for the entire database. *DATES* is then shortened by compression according to the user's input. While this procedure may appear inefficient, it is not significantly more expensive than alternative methods and does permit maximum flexibility in specifying the required timeframe.

Cost Cutter No. 4: I have found that *TIMEFRAME* is usually only activated once per workspace session. Accordingly, care should be taken to ensure that it is not called by other functions which will be activated more frequently.

The cost of running *TIMEFRAME* is somewhat dependent on the length of the specified time series. For instance, setup costs for a 48 day timeframe are \$0.50/\$0.40/\$0.28. A 322 day timeframe will cost about \$0.72/\$0.56/\$0.40.

So far, we have specified the *RETRIEVE* state settings. Moreover, we have defined a global variable *DATES* which represents a vector of dates corresponding to the data points available in the database and as modified by our constraints.

We next extract the data using the function *GETDATA*. *GETDATA* can be generalized for use with any database. However, for this example we have coded it specifically to select open, high, low, close, and volume statistics for each stock code identified in the input. In analyzing the structure of *GETDATA*, the following should be noted:

- 1) The function prompts for a stock code or codes, but does not provide for anything except a very rudimentary input check.
- 2) Correct entry is defined by database convention and the actual stock codes. If we wish to extract information on Amdahl Corporation which trades on the American Stock Exchange, the correct input would be *AAMH*. If, in addition, we require information on IBM (New York Stock Exchange) and Dome Petroleum (Toronto Stock Exchange), we would enter *AAMH, NIBM, TDMP*.

- 3) Using database fact codes (identifiers), we select the open, high, low, close, and volume statistics for the specified stock codes (symbols).
- 4) The required data is now contained in the workspace as global variables (i.e., *NAS3* for fact no. 3, the opening prices).
- 5) Since the required data, in the form of global variables, have the shape of matrices with the first row representing dates, we scrape off the dates, rescale the values since they are stored without decimals, and relabel to variables with logical names (e.g., *OPEN*).
- 6) The workspace now contains redundant variables (the *NAS* variables); these are expunged in order to conserve workspace.

Cost Cutter No. 5: It is possible to evoke a state setting which automatically relabels the global variables to more logical names, but this option adds CPU costs to both the *SETSTATE* and *GETDATA* functions, and should only be used if no further manipulation of the global variables is required prior to processing.

Workspace *TS1* now contains the variables *OPEN*, *HIGH*, *LOW*, *CLOSE* and *VOLUME*. These variables are matrices in which the rows represent stock codes (the actual stock prices) specified by the user. Each column represents a data point (a trading day) specified by *TIMEFRAME*.

The cost of running *GETDATA* depends on the number of facts extracted and, of course, the number of stocks selected. Generally, the cost of activating *GETDATA* is almost totally insensitive to the timeframe specified. For instance, the cost of extracting data for 10 stocks over a 15 month period (322 days at 5 items per stock) is \$11.19/\$8.73/\$6.27, whereas the cost of extracting the same information over a 1 1/2 month timeframe (48 days at 5 items per stock) is \$10.94/\$8.53/\$6.12. A selective exposition of data retrieval costs is given in Appendix 2.

Our information has now been retrieved and is ready for processing or display. Before proceeding with our time series analysis work, it should be noted that it may be advisable to store the raw time series data in a file since the data matrices can get quite large and occupy workspace memory which may be required for filtering or graphics operations.

A review of the retrieval procedure discussed above, in the form of a short terminal session, is presented in Appendix 3.

Checking for errors

I have found the integrity of database information to be remarkably high given the extraordinary number of time series available, the frequency of updates performed, and the fact that data is obtained from a variety of primary vendors. In general the frequency, or rather the infrequency, of errors rules out the necessity of implementing user checks. Errors do however occur and can compromise the user's analysis, especially if database access is performed immediately after updating but before verification by the ultimate vendor. An easy, cost effective solution is available by writing a function which checks for period-to-period fluctuations larger than an arbitrary limit. For example, the daily exchange trading limit for a commodity time series. In coding these check functions, the user should remember that program loops are not necessary and

for cost reasons should never be employed. Simple functions using scans will deliver satisfactory results at low cost.

In practice, the cost of regularly implementing user originated error checks on raw database time series is not, in my opinion, justified.

The moving average as a filter

The selection of useful analytical tools depends upon the behaviour on the time series to be investigated. It happens that the time series of professional interest to me—economic indicators, equity and commodity prices—can be considered to behave more or less like complexly modulated electronic signals. Consequently, my arsenal of investigative tools includes descriptive (and hopefully sometimes predictive) methods of analyzing frequency, amplitude and phase displacement.

At this point I would like to introduce the concept of a filter (also known as a data filter, numeric filter, or digital filter). Conceptually a filter can be thought of as an algorithm, mathematically viewed as a transform, and in practical terms implemented as a computer function. A filter eliminates, or at least dampens, certain information contained in the input time series, producing an output time series which is simpler (contains less information) and is easier to analyze.

The filters I am most familiar with selectively remove frequency patterns embedded in the input time series, although in doing so they also alter the amplitude characteristics of the original data. The frequency response characteristics of a filter are subject to the fundamental design of the filter and, in most cases, local state settings which can be specified for each application. Filter design is a complicated business. The references listed at the end of this paper will provide a comprehensive and practical introduction to this field for interested users.

Today, I will restrict my comments on filters to a brief description of the simplest and most well known one, the unweighted arithmetic moving average. It is constructed by selecting an integer smaller than the number of elements contained in the input time series. This integer, known as the “term” or “span” of the moving average, is the local state setting which determines the frequency response characteristics of this primitive but highly effective filter. Assuming a span of, say, 15; the first 15 elements of the input time series are added then averaged by dividing by 15. Then, the second 15 elements (2 through 16) are added and averaged, and so on, until the input series is exhausted. The output series thus generated represents the filtered product.

In designing filters, the user must take particular care to adhere to efficient APL programming practices since there is often a tendency to use looping algorithms which originate in other programming languages. A looping algorithm for this type of application is probably the most inefficient use of APL imaginable. In these cases simple mathematical operations are performed repetitively forcing the APL interpreter to do far more work than is necessary. Usually a combination of scan and other simple APL operators will achieve the same objective at very little cost.

As an example the function *MOVAVG* or *MOVAVGDOC* (See Appendix 1) accepts as arguments a vector (V) representing raw (time series) data and a single integer (M) which represents the span of the unweighted moving average. The result is a vector of length $M + 1 + \rho V$.

The value of M will determine the frequency response of the filtered output. Large values for M will result in a smoother output vector; that is, it will eliminate the higher frequencies.

The interpretation of filtered product is an interesting and complex field which is beyond the scope of this paper. However, as indicated above, the publications listed in the reference section of this paper are highly recommended as providing a sound introduction to the subject.

Appendix 1

Listing of Functions

```

▽ GETDATA;FACTS;JUNK
[1] CODE←PROMPT 'CODE? ' ◇ →0 IF 0=ρ, CODE
[2] FACTS← 3 4 5 6 9 ASELECT NASTOCK FACTS I.E. OPEN HIGH LOW CLOSE VOLUME
[3] JUNK←CODE ΔGET FACTS A WS NOW CONTAINS DATA IN VARS <<NAS3,NAS4,NAS5,NAS6,NAS9>>
[4] OPEN←0.001× 1 0 +NAS3 ◇ HIGH←0.001× 1 0 +NAS4 ◇ LOW←0.001× 1 0 +NAS5
[5] CLOSE←0.001× 1 0 +NAS6 ◇ VOLUME← 1 0 +NAS9
[6] A STRIP OUT REDUNDENT DATES AND RESCALE AS APPROPRIATE.
[7] 6 □FD 'NAS3 NAS4 NAS5 NAS6 NAS9' A EXPUNGE REDUNDANT VARS.
▽

▽ R←M MOVAVG V
[1] R←(÷M)×(M-1)+R-(Mρ0), (-M)+R+÷V
▽

▽ R←M MOVAVGDOC V;A;B
[1] R←÷V
[2] A←(-M)+R
[3] B←(Mρ0), A
[4] R←R-B
[5] R←(M-1)+R
[6] R←R÷M
▽

▽ SETSTATE
[1] ΔVIEW 'NASTOCK'
[2] ΔSTATE← 0 0 0 1 1 0 0 0 0 0
[3] A
[4] A INITIALIZE WS USING FNS IN WS 55 RETRIEVE.
[5] A ACTIVATE MANUALLY OR BY □LX.
[6] A SEE FN SETSTATE2 FOR ALTERNATE WAY.
▽

▽ SETSTATE2
[1] ΔVIEW 'NASTOCK' A SPECIFY STOCK MARKET DATABASE.
[2] ΔERRORTYPE 'EXPLICIT' A RETURN ERROR MESSAGES
[3] ΔRESULTTYPE 'GLOBAL' A AS OPPOSED TO 'PACKAGE'
▽

▽ TIMEFRAME;NONE
[1] DATES←~190000000+Φ'DAILY' ΔSEARCH10
[2] A SPECIFY FREQ., SELECT ALL DATES, ROTATE, SUBTRACT CENTURE
[3] NONE←(ρDATES)ρ1 A DEFINE A SELECTION VECTOR FOR ALL DATES
[4] START:'ENTER DATE CONSTRAINTS. EG DATES≥810115 OR (DATES>811201)∧DATES≤820103 (YYMMDD)'
[5] 'ENTER <NONE> IF YOU HAVE NO CONSTRAINTS *** WATCH OUT***, <NONE> CAN BE EXPENSIVE'
[6] →NEXTL IF(ρDATES)=ρINPUT+□ ◇ 'LENGTH ERROR' ◇ →START
[7] →NEXTL IF÷/INPUT ◇ 'YOUR CONSTRAINTS HAVE SELECTED NO DATA - REENTER' ◇ →START
[8] DATES←INPUT/DATES AREDEFINE DATES USING INPUT AS SELECTOR FOR COMPRESSION.
[9] 'DAILY' ΔTIME DATES+19000000 A SPECIFY TIMEFRAME FOR FACT SEARCH
▽

```

Appendix 2

Selected Database Retrieval Costs (expressed in T-task costs)

TIMEFRAME: 15 months (322 days) Retrieve 5 data items/day/stock

	1 Stock	5 Stocks	10 Stocks
SETSTATE	0.35	0.35	0.35
TIMEFRAME	0.72	0.72	0.72
GETDATA	2.24	6.26	11.19
	-----	-----	-----
	—	—	—
Total	\$3.31	\$7.33	\$12.26
No. of Data Items	1,610	8,050	16,100
Cost/100 Data Items			
T-task	\$0.21	\$0.09	\$0.08
N-task	\$0.16	\$0.07	\$0.06
B-task	\$0.12	\$0.05	\$0.04

TIMEFRAME: 1 1/2 months (48 days) Retrieve 5 data items/day/stock

	1 Stock	5 Stocks	10 Stocks
SETSTATE	0.35	0.35	0.35
TIMEFRAME	0.72	0.72	0.72
GETDATA	2.19	6.03	10.94
	-----	-----	-----
	—	—	—
Total	\$3.26	\$7.10	\$12.01
No. of Data Items	240	1,200	2,400
Cost/100 Data Items			
T-task	\$1.36	\$0.59	\$0.50
N-task	\$1.06	\$0.46	\$0.39
B-task	\$0.76	\$0.33	\$0.28

Appendix 3

Terminal Session Illustrating Retrieval Operations

```

)LOAD TS1                                ...load TSI on 82.04.12
SAVED 0.25.58 04/12/82

SETSTATE                                ...activate SETSTATE
TIMEFRAME                              ...activate TIMEFRAME
ENTER DATE CONSTRAINTS, EG DATES≥810115 OR (DATES>811201)^DATES≤820103 (YYMMDD)
ENTER (NONE) IF YOU HAVE NO CONSTRAINTS *** WATCH OUT***, (NONE) CAN BE EXPENSIVE
□:
    DATES≥820402                        ...specify date constraints
    GETDATA                            ...activate GETDATA
CODE?
    AAHM,NIBM,TDMP                    ...select stocks
                                        ...retrieval was successful information
                                        is now in workspace.

    DATES
820402 820405 820406 820407 820408

    OPEN
20.5    21.5    20.75    21    21.25
61.25   61.5   61.75   61.75  61.75
8.625   8.875  8.5     8.5    8.875

    HIGH
21.25   21.75   21.125  21.5   21.25
61.875  61.75   61.875  62.125 62.5
8.875   8.875   8.626   9     9.875

    LOW
20.5    20.75   20.75   20.875 20.625
61.125  61.25   61     61.5   61.75
8.375   8.375   8.25   8.5    8.875
                                        ...display variables

    CLOSE
21.25   20.875  21.125  21     20.75
61.5    61.5    61.625  61.75  62.375
8.875   8.375   8.5     8.75   9.75

    VOLUME
54700  26300  18900  28700  9000
674700 399400 404200 495100 465200
120518 71525 126055 150287 349419

```

Appendix 4

Selected Annotated References

1. Cleeton, Claud E. *The Art of Independent Investing*. Prentice-Hall Inc., 1976.

This publication is largely an update and in some cases an amplification of Hurst's earlier work.

2. Davis, John C. *Statistics and Data Analysis in Geology*. John Wiley & Sons, 1973.

Davis's work (including FORTRAN programs by R. J. Sampson of the Kansas Geological Survey) constitutes a comprehensive review of the tools available to investigate the nature of geologic data populations. At the same time it remains one of the most readable and useful general statistics textbooks available. Of particular interest is chapter 5, Analysis of Sequences of Data, which describes interpolation procedures, runs tests, regression techniques, filtering and time trend analysis, autocorrelation and cross-correlation, cross-association of qualitative sequences, and a section on frequency domain analysis. This publication is highly recommended.

3. Hurst, J.M. *The Profit Magic of Stock Transaction Timing*. Prentice-Hall, Inc., 1970.

Notwithstanding the promotional nature of this book's title, Hurst's work represents somewhat of a milestone since his "systems" approach to stock market transactional analysis inspired a complete re-examination of static methods which had been in use for over 50 years. Hurst considers a stock time series as being similar to a complexly modulated signal to which can be applied the same analytical tools used to dismember and reconstruct electronic signals. Of particular interest is his use of multiple moving averages with moving average spans specifically selected to give graphic approximations of the frequency domain characteristics of the stock signal in question. Moreover, his work in the frequency domain outlines definite periodicities, not only in specific stock signals but also in widely quoted measurements such as the Dow Jones Industrial Average.

4. Kaufman, P.J. *Commodity Trading Systems and Methods*. John Wiley & Sons, 1978.

This excellent publication, obviously heavily influenced by the author's personal experiences in speculative commodity trading, contains a wealth of information on commodity trending models, the practical application of moving average systems in making business decisions and a treatment of various momentum indicators and oscillators. Moreover, his sections on behavioural techniques and time series pattern recognition provide insight into time series analysis from the intuitive, rather than the quantitative, point of view.

5. Kaufman, P.J. (Editor) *Technical Analysis in Commodities*. John Wiley & Sons, 1980.

This publication records papers given during a symposium on "Technical Analysis in Commodities" held in New York in October 1978. This publication con-

tains two unusually interesting papers. In "Selection and Direction", J. Welles Wilder, Jr. describes a trend index (applicable to, say, commodity, stock, and currency time series) which generates short term predictive signals, often even if the time series is not strongly trending in any particular direction. Of even greater interest is Frank Hochheimer's paper on "Moving Averages". It summarizes Hochheimer's research into the computerized testing of simple, linear, and exponentially smoothed moving averages in the context of their predictive qualities as applied to actual trading situations. His major conclusion, a controversial one no doubt, is that in most cases the simple moving average works better than either the linear or the exponential average.

6. Wheelwright, Steven C., and Spyros Makridakis. *Forecasting Methods for Management*. John Wiley & Sons, 1973.

This reference bridges the theoretical and practical aspects of time series analysis. It is an excellent source book in terms of its presentation of data smoothing techniques, especially exponential moving averages and various adaptive filtering techniques.

MONITORING OF INTERNATIONAL EXPOSURES

Mark Fuller, Assistant Manager
International Division
Commonwealth Trading Bank of Australia
Sydney, Australia

Due to rapid expansion into international operations since the mid-1970s, the Commonwealth Trading Bank of Australia (CTB) decided several years ago to develop a computer-based system for the monitoring of international exposures. This system, known as the Global Limits System, makes use of the I.P. Sharp computer situated in Canada and allows potential live access to the computer by a number of dealing centres throughout the world.

Limits are determined at the CTB's Head Office for dealings with both banks and countries. These limits are divided into five categories of varying risks, namely:

- **Deposits:** This limit relates to deposits placed with other banks for fixed terms of less than 12 months.
- **Foreign exchange:** This limit relates to spot and forward foreign exchange deals (aggregate of purchases and sales) with banks.
- **General:** This limit relates to exposure for general operations, most of which arise from normal trade transactions and includes:
 - Confirmation of overseas banks' letters of credit.
 - Drawings negotiated/paid/accepted under overseas banks' letters of credit where the CTB does not have a right of recourse on the credit beneficiary.
 - Short term overdrawings on overseas banks' accounts conducted with the CTB.
 - Risk participations where an overseas bank offers some of its risk with certain banks to the CTB.
 - Bankers acceptance facilities.
- **Commercial loans:** This limit relates to longer term, direct exposure.
- **Other:** This category is used for items which do not fall into any of the above categories.

Each of the above five categories of limits is sub-divided into five time periods—0-2,

3-7, 8-180, 181-365 and 365+ days. This enables limits to be reduced (if desired) as maturity lengthens.

Another of the types of limits that transactions are checked against is the credit risk limit (CRL). This concept is a prudential control on the amount of transactions maturing with any bank within any seven-day period on either side of its maturity date. This limit would normally be greater than (or equal to) the spot limit but less than (or equal to) total limits for the bank.

A further type of limit is the country limit. The concept of a country limit is the maximum amount that we wish to be exposed to that country. This limit would be expected to be equal to, or less than, the sum of the individual deal category limits within that country.

As each deal is input, it is checked against these limits and a warning message issued if an excess occurs.

The system allows for various information to be included about any country, bank or deal. This may be in the form of a deal subtype (normally used in those categories where there is some differential required as to the type of transaction—that is, “General”, “Commercial Loans” and “Other”—bank short name and country of head office, banks with head offices within a certain country, special restrictions for banks and countries, etc.

Special restrictions can be placed on any bank or country and may indicate a variety of information. This may range from a general message, for example, *THIS BANK/COUNTRY IS A MEMBER OF THE COMECON GROUP OF BANKS/COUNTRIES*, to specific instructions such as *DEPOSITS RESTRICTED TO OVERNIGHT*.

All programs written on the system have user assistance messages should the operator have difficulty while at the terminal. These messages are precise and simple and are designed to guide the operator through any program without the aid of a user manual.

To aid dealing centres further, a mirror image system has been designed (called the test/demonstration system) in which a dummy set of banks has been created. These files can be added to, deleted, or amended in any way that the real system can. In short, operators using the test/demonstration system may use *all* programs available, including those normally restricted to Head Office only (e.g., setting of limits), and, secondly, data on the test/demonstration system can be changed in any way without damaging the live system.

The following are the main features of the Global Limits System:

1. AUTOMATICALLY GENERATED REPORTS

A number of reports are printed automatically at each centre each day following the first sign-on for the day. For the purpose of this paper, we have assumed that the CTB's Head Office is in Australia and that the local centre used in various examples is situated in the U.S.A. Please note also that the currency applicable to Australia is Australian dollars, written as *AUD*, and in the U.S.A. is United States dollars, written as *USD*.

These reports are:

- (a) *Deals Added Report*: Lists all deals added to the system on the previous working day by that centre. Full details of the program used to add deals to the system are shown in Section 2.

An example of this report is seen below.

BANK NO	BANK SHORT NAME	DEAL TYPE	COUNTRY	CURRENCY	FOREIGN CURRENCY AMOUNT	CENTRE CURR AMOUNT	DOCKET NO	DEAL ID NUMBER	MATURITY DATE
2103	ABC BANK	2 FWD	142SGP	36SGD	450,000	215,763	39293	134944	030683
4187	DEF BANK	1 DEP	172USA	10USD	1,000,000	1,000,000	62001	134945	101283

The examiner can therefore see that his centre has completed a forward foreign exchange deal with ABC Bank (no. 2103), Singapore Office (no. 142) for Singapore \$450,000 due to mature 3 June 1983. He can also see that the dealing docket number was 39293 and that the equivalent of the deal in his centre's local currency is U.S. \$215,763.

- (b) *Deals Matured and Automatically Deleted Report*: On the day that deals mature they are automatically deleted from the system. This report lists all such deals.

An example of this type of report is seen below.

BANK NO	DEAL TYPE	COUNTRY	CURRENCY	FOREIGN CURRENCY AMOUNT	CENTRE CURRENCY EQUIV	DEAL ID NO	ENTRY DATE
217	2 SPT	142SGD	24ITL	400,000,000	313,619	134829	041082
1210	1 DEP	113NZL	30NZL	1,600,000	1,227,050	135702	110682
1388	1 DEP	175USA	10USD	2,000,000	2,000,000	142306	210682

- (c) *Limit Excess Report*: Lists all limit excesses incurred by that centre. Head Office receives a master copy of this report, which shows excesses by all centres.

An example of the master excess report that Head Office would receive is shown below.

BANK NO	BANK SHORT NAME	LIM TYPE	LIMIT	DEAL AMOUNT	TOTAL EXP FOR THIS LIMIT	EXCESS IN AUD	MATURITY DATE	CENTRE
1234	RIVER BK	DEP	3,000,000	4,000,000	4,000,000	1,000,000	121282	NY
1234	RIVER BK	DEP	3,000,000	2,000,000	6,000,000	3,000,000	211282	LA
2345	STORAGE BK	FWD	20,000,000	10,000,000	23,750,000	3,750,000	100383	HK

- (d) *Changes Report*: This is generated at all centres whenever there has been a change to the system, e.g., a change in limit. It will also alert or remind centres of the downgrading or cancellation of limits, where such is considered necessary by Head Office, and of special restrictions which may be established or deleted.

Examples of entries on this report are found below.

(A)	NEW BANK	4296	JAPANESE BK	ADDED AT NY
(B)	BANKSET	1732	XYZ BANK NOW 71	WAS 251
(C)	LIMITSET	3860	ABC BANK FRX NOW:	1,000,000, 2,000,000, 2,000,000, 2,000,000, 0
			WERE:	0 , 0 , 0 , 0

Example (a) indicates that a new bank, which has been allocated number 4296 and short name *JAPANESE BK* has been added to the system by the New York office.

Example (b) indicates that *XYZ BANK* (no. 1732), which was previously added to the system, has been allocated to the country Hong Kong (country no. 71). Whenever a new bank is added, it is automatically given country code 251 and once a week the terminal issues a reminder that the banks with this code should be allocated to a country.

Example (c) indicates that foreign exchange limits of spot one million dollars and forward two million dollars (for terms of up to 365 days) have been established. It should be noted that in this example, foreign exchange limits have not been allocated for terms exceeding one year.

2. ADD

ADD is a program used to add deals to the Global Limits System. For each deal the following information is input:

- Bank with which deal was transacted.
- Transaction type—i.e., deposit, foreign exchange, etc.
- Country of bank office with which deal was transacted.
- Currency of transaction.
- Transaction amount.
- Transaction reference number.
- Maturity date of the transaction.
- Subtype (optional).

All of this information is input to the system by use of numerical codes.

An example of CTB's input would be:

1234	175	2	10	1,000,000	12345	310183
BANK	COUNTRY OF	FOREIGN	CURRENCY	AMOUNT	TRANS	MATURITY
NO	BANK'S	EXCHANGE	CODE		REFERENCE	DATE
	OFFICE	TYPE DEAL			NO	(31/1/83)

All deals are stored in their original currency amounts and for information purposes are converted to the local currency of each centre at the mid-rate (average of buying and selling rate) before exposure information is received. In this way, all centres using the system receive limit and exposure in their local currency.

3. *FREE*

This is a program designed to show how much scope remains to deal with a particular bank. The operator merely inputs the bank number, the deal type and the time period required (more than one time period may be input). The system will then show the limit for each time period selected followed by the amount free to deal.

An example of input would be:

5144 2 180

and the printout would be:

5144 TRADING BANK LTD 180 DAYS FORWARD 15,000,000/6,000,000

indicating that for bank 5144 in the 180 day forward category, the forward limit is \$15,000,000 and the amount free to deal without exceeding the limit is \$6,000,000. This process takes only seconds and does not delay the dealer's response.

4. *DEALS*

This program provides a listing of all deals in the Global Limits System which satisfy certain requirements specified by the operator.

DEALS is a particularly powerful and widespread program which the user can utilize to opposite extremes to obtain a printout of one deal or all deals on the system, or any specified amount between the two. For example, one can determine all the outstanding deals by any centre for a certain maturity spread, or more specifically, the outstanding deposits placed with a certain bank on a certain day.

Below is an example of this program.

Input could be:

: *BANK* 5393
: *TYPE* 3

We are asking for all deals of type 3 (general transactions) with bank no. 5393, and the printout would be:

<i>BANK</i>	<i>DEAL TYPE</i>	<i>SUBTYPE</i>	<i>COUNTRY</i>	<i>CURRENCY</i>	<i>FOREIGN CURRENCY AMOUNT</i>	<i>CENTRE CURRENCY EQUIV</i>	<i>ENTRY DATE</i>	<i>MATURITY</i>	<i>DEAL ID DATE</i>	<i>CENTRE NO</i>
5393	3	9	127PHI	99AUD	222,428	234,839	270582	311182	134487	SYD
5393	3	13	175USA	10USD	5,000,000	5,000,000	030882	290183	146231	NY

Subtypes referenced:

9—confirmed letter of credit
13—direct bankers' acceptance

5. *BANK*

This program can provide various information about a selected bank including country of its head office, total limits and exposure, maximum exposure during a given time-span, up-to-the-minute exposure details, the number of deals transacted by each centre and any special restrictions imposed on dealing with that bank.

Below is an example of the printout.

5144 JAPANESE BANK LTD H.O. 83 JAPAN CRL 10,000,000							
NO SPECIAL RESTRICTIONS FOR THIS BANK.							
<i>DEAL</i>	<i>TOT EXP TYPE</i>	<i>DAYS</i>	<i>2</i>	<i>BANK LIMITS IN '000'S</i>			
				7	180	365	9999
<i>DEP</i>	1,894,298	15,000	15,000	15,000	15,000	15,000	0
<i>SPT</i>	0	1,000	1,000	0	0	0	0
<i>FWD</i>	909,001	0	0	1,000	1,000	1,000	0
<i>GEN</i>	121,024	17,000	17,000	17,000	17,000	17,000	17,000
<i>C/L</i>	0	0	0	0	0	0	0
<i>OTH</i>	0	0	0	0	0	0	0
<i>TOT</i>	2,924,323						

This shows that Japanese Bank Ltd. has total exposure of \$2,924,323 (U.S.). One can also see that deposit exposure is \$894,298 (U.S.), spot exposure is nil, forward exposure is \$909,001 (U.S.), general exposure is \$121,024 (U.S.) while commercial loan and other exposure is nil.

6. COUNTRY

The program *COUNTRY* prints information about a selected country including country limit, bank numbers of banks with head offices in that country, any special restrictions applying, limits for each deal type divided into each of the five time categories, and current exposure broken up on a centre-by-centre basis. This is shown in the example below.

: C 171									
171 ZYX COUNTRY									
BANKS WITH HEAD OFFICE IN THIS COUNTRY ARE NUMBERED -									
2316, 2714, 2681, 4809, 4993, 5874									

DEAL	TOT EXP	DAYS	2	SUM OF BANK LIMITS IN '000'S					
TYPE				7	180	365	9999		

DEP	21,825,351		24,000	24,000	24,000	0	0		
SPT	60,120,837		61,000	0	0	0	0		
FWD	139,266,829		0	293,000	293,000	293,000	0		
GEN	450,789		10,000	10,000	10,000	10,000	0		
C/L	3,628,890		4,484	4,484	4,484	4,484	4,484		
OTH	0		3,000	3,000	3,000	3,000	3,000		
TOT	225,292,696								

SYD/PM	NO	LONDON	NO	NY/GC	NO	HONG KONG	NO	LOS ANGELES	NO
	DLS		DLS		DLS		DLS		DLS

0	0	947,159	0	0	1	20,878,192	86	0	4
2,344,244	108	2,992,423	71	9,000,000	53	43,576,470	65	2,207,700	2
31,536,250	11	50,034,988	32	57,695,591	31	0	0	0	0
450,789	8	0	0	0	0	0	0	0	0
3,628,890	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
37,960,173	130	53,974,570	103	66,695,591	85	64,454,662	151	2,207,700	6

You will note the columns listing number of deals on a centre-by-centre, deal type-by-deal type basis. This represents the total number of deals transacted by each centre since a certain specified date. This aids in the reviewing of limits, and in other decision-making processes.

General

Programs are included in the system to set limits and other information relating to banks and countries and to set exchange rates. A password is required in order to obtain access to these programs which ensures that these matters are controlled by a small number of people within Head Office.

We believe that the Global Limits System is one of the most advanced means of monitoring exposures that is available at present and we are, therefore, willing to consider the sale of the software package to interested parties. Any enquiries should be directed to:

The Chief Manager, International
Commonwealth Trading Bank of Australia
G.P.O. Box 2719
Sydney, N.S.W. 2001
Australia
Telex 20072, 25322

DEVELOPING APL SYSTEMS AT THE URBAINÉ LIFE RE

Michal I. Bauer
Assistant Vice President and Actuary
The Urbaine Life Reinsurance Company
New York, New York

Introduction

The Urbaine Life Reinsurance Company is a young and aggressive professional life reinsurance company, chartered in New York in 1975. The basic operating functions of the company were initially set up on manual systems, and the process of converting to computerized administration is just getting under way.

About two years ago we started using SHARP APL for certain actuarial operations. Since we are a young and fast growing company, everyone gets to wear many hats in the course of doing his or her job. My hats include all actuarial functions and related data processing work, among others. As a result, programming financial systems is not something I can afford to spend a large chunk of time on.

The nature of a reinsurance company must be such that it be able to react very quickly to demands made by its clients. Reinsurers almost never have the kind of lead time to price new products that direct insurers have, because the direct insurers usually request reinsurance pricing when most or all of their pricing work has been completed. At that time they often need to know the reinsurance cost in order to finish their pricing, and sometimes they have begun selling the new product before establishing specific reinsurance support.

All of this is normal and to be expected in reinsurance. But it means that, if you work for a reinsurance company, your time is frequently chopped up by new assignments that have just come up and need to be handled immediately. In other words, it's almost impossible to dedicate even a week or a few days to write a system that you know you need to write. I am sure that many actuaries working at direct writing companies, especially small ones, and certainly many consultants, are familiar with this kind of situation. Given this environment, I have been drawn by the circumstances of my job to taking what I would call an "organic" approach to developing the systems that we have needed. These systems include a valuation system, a 6-year financial projection model, and a YRT pricing and rate printing system.

Keeping all of the above in mind, I will discuss the nature of the "organic approach" to systems development, and its advantages and disadvantages. As an extension to and an example for this discussion, I would also like to describe Urbaine's use of word

processing equipment to perform Input/Output Operations within the systems that we have developed.

The “organic approach”

What do I mean by an organic approach?

Webster defines organic as a “systematic arrangement of parts into a whole”. Simply put, it could be called the opposite of top-down programming—or bottom-up programming. The initial programs or functions written perform the procedures that are most urgently needed. For example, in a valuation system, the functions that calculate the reserve factors would be written first. Then if that’s all the time available, at least that much can be put to use in doing the valuation. The next time the programmer can work on it, he or she can add the function that is most urgent at that point.

In this fashion, a reasonably complex system can be “grown” by writing the parts and systematically building them up to form the whole. With each step the amount of manual summarization and data preparation decreases and the proportion of automation increases.

Documentation of the system grows naturally with the system because each function can be written with internal commenting. Ultimately, when the system is complete, most of the commenting can be removed from the functions and written into a system manual.

What are the advantages to using this approach of “growing” data processing systems? First, this method adapts itself especially well to schedules which do not allow the programmer to dedicate him or herself to working only on the system. Where it is possible for a programmer to so dedicate his or her time, normal system development techniques may ultimately produce a more efficient system. Second, it allows the user to get the most output from the least amount of work, since each function is directly usable as soon as it gets written, and actually serves to enhance the portions already developed. Third, different modules of the programming can easily be farmed out to other technicians who are already familiar with the particular procedure that they are to program. By technicians I mean, for example, actuarial students—not programmers specifically. Fourth, system changes are fairly easy to accommodate up until the time when a “master program” has been written. The author of the system has the opportunity to operate it at various stages of its development, and can therefore, through experience, put the pieces together in the most efficient manner. In volatile business situations, when things change frequently and not too predictably, this kind of flexibility can be extremely valuable. Fifth, should the system become obsolete for any reason, it would be comforting to know that there was actually a minimum amount of time directly invested in the system, that it was being used during the time of its development, and that there might be portions of it that would still be usable in the new order of things. And finally, the transition from manual to computerized operation is a gradual one, where each function is thoroughly debugged and understood by the user by the time it is merged into the total system.

Beginnings of automation

At this point I'd like to move on to describe how this principle has led to Urbaine's use of word processing equipment to communicate with the SHARP APL network.

Prior to two years ago, Urbaine was not directly using any computers or word processors. Then approximately two years ago, we acquired a pair of Lanier "No Problem" word processors to be used in the Administration area. Administration immediately began transferring in-force files to word processing disks, so that reinsurance in-force lists could be efficiently and attractively generated for our clients. Valuation input listings were soon created from those disks.

Shortly thereafter the Financial area subscribed to I.P. Sharp, mostly for actuarial purposes. That is when the automated valuation system was begun. Each quarter we add functions that simplify the work that is left for us to do. At one point last year our biggest problem became inputting new business into the valuation files. It was clear that the next step in the valuation process was to establish an automated link between the Administration and Financial areas, so we purchased Lanier's Communication Option.

Communicating with word processing equipment

The Communication Option, in order to function, requires some hardware attached to the word processor, a telephone connection, and two programming disks. The procedure is simple. First a TTY-ASCII programming disk, provided by Lanier, is loaded into the word processor, followed by another disk, provided by I.P. Sharp, which loads a parameter table to enable the word processor to function in SHARP APL. A simple command puts the processor into communications mode, a phone call is placed, and the standard sign-on procedure is used to connect the processor as a terminal. Once the connection is made, the operator can insert additional disks to either send or receive data to or from the SHARP APL network. At any point during the on-line session the operator can switch to the on-line editor mode, maintaining the communications link, and make any word processing changes that are needed.

Acquiring the Communications Option was a major step forward in automating the valuation process. However there were a few problems along the way which have been for the most part worked out, but which are worthy of mention.

The first problem was the large number of transmission errors that resulted when we were transmitting data from word processing disks to APL files. We discovered that the word processor connection seemed to be highly sensitive to static interference. The simplest solution seems to be to do all of our Lanier to I.P. Sharp transmissions after 4:30 or 5:00 in the afternoon when the static no longer gets in the way. (Interestingly enough, this problem does not occur when we are transmitting from I.P. Sharp to Lanier.)

Another problem resulted from the fact that while the word processor can accept special APL characters from the computer, it cannot accept them from the keyboard. Sometimes it is necessary to manipulate data while working on the word processor, although usually manipulations are much more easily done on a terminal. The solution is to use a set of small functions that eliminate the need for special characters. The following are some examples:

Figure 1

```

      ▽ W APPEND T
[1]   W □APPEND T
      ▽

      ▽ ASSIGN V
[1]   R←V
      ▽

      ▽ R←DIM SHAPE M
[1]   R←DIMpM
      ▽

      ▽ SIZE TC
[1]   □SIZE TC
      ▽

      ▽ R←RC TAKE M
[1]   R←RC↑M
      ▽

```

Transmitting data from the SHARP APL Network to the word processor is relatively easy. The word processor has to be put on automatic memorization, so that as each chunk of information equal to one page is transmitted, it is automatically memorized onto a disk and the next page begins to accept data. The entire terminal session is memorized, including any display of variables or file components. At the end of the transmission the disk can be edited in word processor mode to contain only what is needed.

Because transmitting from Lanier to I.P. Sharp is a bit more complicated, I.P. Sharp provided us with a pair of functions to do the transmissions. The first one, *TRANSMIT*, converts word processing data to character strings and transmits the data into distinct file components. It is shown below:

Figure 2

```

      ▽ TRANSMIT;V;AA;R
[1]   □UNTIE □NUMS
[2]   'TRANSFILE' □STIE R←QT
[3]   □PW←250
[4]   AA←□AV[□IO+1]
[5]   ' TRANSMIT ITEM : '
[6]   L1:AA←AA,AESARBTYP[1+(~V∈ 17 19)/V←□ARBIN 10 7] ⋈ IGNORE
      ×ON (17) AND ×OFF (19)
[7]   →((( '$',□AV[157])v.≠-2↑AA)^( '≤',□AV[157])v.≠-2↑AA)/L1
[8]   AA □APPEND R
[9]   L2:□PW←132
[10]  □ARBOU 10 ◇ ' END OF ITEM. '
      ▽

```

The second function, *CONVERT*, converts the character strings into numeric matrices and translates specific alpha-code into numeric code. It is shown below along with the additional functions and variables it requires:

Figure 3

```

▽ R←CONVERT ITEM;□IO
[1]  A FUNCTION NEEDS GLOBALS CR; SYMB AND SUB FUNCTIONS ΔBOX AND ΔREPLACE
[2]  □IO←1
[3]  ITEM←(∼ITEM=',')/ITEM AREMOVE ALL COMMAS
[4]  ITEM←(∼ITEM='+')/ITEM AREMOVE ALL DASHES. DASHES APPEAR AS '+'
[5]  ITEM←ITEM ΔREPLACE '|1' AREPLACE SYMBOL FOR <M> WITH A 1
[6]  ITEM←ITEM ΔREPLACE '|_2' AREPLACE SYMBOL FOR <F> WITH A 2
[7]  ITEM←ITEM ΔREPLACE '|L1' AREPLACE ALL OCCURRENCES OF <L> WITH A 1
[8]  ITEM←ITEM ΔREPLACE '|1' AFOLLOWING 3 LINES WILL CONVERT SYMBOL <|~|> TO 100
[9]  ITEM←ITEM ΔREPLACE '~0'
[10] ITEM←ITEM ΔREPLACE '|0'
[11] ITEM←ITEM ΔREPLACE '□2' AREPLACE ALL OCCURRENCES OF CAPITAL <L> WITH A 2.
[12] ITEM←ITEM ΔREPLACE 'τ1' AREPLACE ALL OCCURRENCES OF <N> WITH A 1.
[13] ITEM←(∼ITEM∈SYMB)/ITEM AREMOVE ALL NON-NUMERIC CHARS.
[14] A FOLLOWING TWO LINES CONVERT ITEM TO A NUMERIC MAT.
[15] R← 0 7 ρ0 ◇ ITEM←CR ΔBOX ITEM ◇ ITEM←(∼ITEM^.=' ')/ITEM
[16] L:R←R,[1] 7†‡ITEM[1;] ◇ →(0∈ρITEM← 1 0 †ITEM)/0 ◇ →L

```

▽

```

▽ R←A ΔBOX B;C;D;E
[1]  →(2=ρR←B)/0
[2]  ‡((R=0)=□NC 'A')/'A←'',''
[3]  E←0[[/D← 1+D←D-0, 1+D←(C+B∈A)/1ρB←B,A
[4]  R←((ρD),E)ρ(,D°.≥1E)\(∼C)/B

```

▽

```

▽ OBJ←OBJ ΔREPLACE PARTS;BV
[1]  A FUNCTION REPLACE IN OBJ THE FIRST ELEMENT OF PARTS
[2]  A WITH THE SECOND ELEMENT OF PARTS
[3]  OBJ[(OBJ=1+PARTS)/1ρOBJ]← 1+PARTS

```

▽

CR

(The variable CR represents a carriage return.)

SYMB←(∼□AV∈ '0123456789 ',CR)/□AV

This function was developed for a typical format of our input. It is easily modified to use on other formats.

Uses of word processor communications at Urbaine

The word processing link has been a very valuable one for the Urbaine. We have used it predominantly in the valuation process for inputting and correcting valuation files.

We have also used it in the rate-making process, to print special YRT rates for special situations. Printing on the Lanier printer produces letter-quality rate sheets (as compared to our dot-matrix terminal) with much easier formatting, as all headings and other adjustments are easily done in word processor mode.

We have not yet used the word processing communications link in our Projection system, but we foresee that one of the next developments in that system will be letter quality exhibits automatically produced through this link.

Conclusion

The "organic approach to systems development" is a phrase I have coined to describe a bottom-up approach to automating existing manual systems. It is particularly helpful and appropriate to situations where technicians who are not professional programmers write systems to automate jobs they are currently doing manually and do not have the kind of schedules that allow them to devote themselves entirely to systems development for any extended period of time. This is sometimes the case for actuaries working at small companies or on their own.

One of the by products of this approach at the Urbaine has been the establishment of a communications link between the SHARP APL Network and Lanier word processors, as a natural extension of the previously existing input/output procedures. Urbaine's uses of this communications link are continuously expanding as we further automate and coordinate our administrative and actuarial systems.

GOVERNMENT BUDGETING: A FLEXIBLE SOLUTION TO A LARGE PROBLEM

**David Ford
Special Projects Analyst
Ministry of Finance, Treasury Board Staff
Province of British Columbia
Victoria, British Columbia**

The problem

Budgeting expenditure in a government context is very difficult to do well. There are a number of reasons why this is so. Governments tend to be large in size, and to be very complex in their organization and in the functions they carry out. Also, the level of revenue available in a given fiscal period is more fixed than it is in a private enterprise or, to put this point another way, the link between the level of revenue and the level of expenditure is not as direct, so correctness in the annual expenditure figures is very important.

The Province of British Columbia, where I work, has all of these characteristics. It employs some 40,000 people directly, and probably another 180,000 indirectly through grants to hospitals, school boards, universities, and so on. Its operating budget for this fiscal year is approximately 7.6 billion dollars. It has within it 23 ministries and offices, which vary in size from the Ministry of Health, with a budget in fiscal 82/83 of \$2,236,586,249 to the Office of the Premier, with a budget of \$709,124. Some of the larger ministries are very decentralized in their operations. The Ministry of Human Resources, for example has around 1,700 individual budget centres. Most ministries perform a wide variety of functions. The Ministry of Lands, Parks and Housing, for instance, manages all publicly owned land in the province (approximately 98% of the total), looks after all provincial parks, and is involved in residential and urban development. These are just its main functions. The bulk of the province's revenue comes from personal and corporate income, sales and gasoline taxes, and from the federal government, and none of this revenue is directly generated through its spending activities.

It is, to say the least, a challenge to develop a budgeting system that will serve the needs of an organization such as this. Besides coping with a good deal of volume and complexity, it must meet a number of other criteria:

- 1) It must be absolutely accurate and trustworthy. The published Estimates in British Columbia are legal documents.
- 2) It must be as economical as possible, both of staff time and of data processing resources, both for system development and operation.

- 3) It must be easy for everyone using it to understand and use.
- 4) It should produce the budget in such a way that actual expenditures can be easily monitored without a great deal of reclassifying and reallocation.
- 5) It should provide significant analytical capability.
- 6) It should be flexible, and permit changes to be made quickly yet in a logical way.
- 7) It must facilitate communication between people who have widely different perspectives on the information the system provides. Cabinet ministers think about the budget in a very different way from front line managers.

If all these criteria are met, then the budget system should make a significant contribution towards making a better budget. By this I mean a budget which gives the best value for money possible to the taxpayer and does this in a way that all managers in the process can feel they have had a real say in its production.

Over the past three years we have been successful in building a budget system which satisfies most of these criteria. Here is how we did it.

The solution

I believe that if we had begun trying to design a system which met all of the criteria I have just outlined using an approved system development lifecycle, the problems involved would quickly have become overwhelming. Such an approach would call for a complete description of the required system before any implementation work began, and it is doubtful whether anyone involved had the intellectual capacity to do this. Perhaps it was partially in recognition of this that the computing services arm of the government suggested a price tag of \$1,000,000 in development costs for the application of their lifecycle to the system.

Instead, the design principle which was followed was taken from *General Systemantics* [Ref. 1]: "Every complex system that works began as a simple system that worked". The design process was interactive or adaptive and on a number of levels. Individual component systems have evolved continuously to meet new user needs and to incorporate various ideas as they came along. New systems have been created to meet newly identified needs (our record development time was three days). The overall system continues to evolve as the needed linkages between the component systems become apparent.

The key to making all of this happen has been the availability of a computer language which permits development (and redevelopment) at a very low cost. The entire system is written in APL.

Of course there are dangers involved in using this evolutionary approach too freely. The level of discipline involved must be very high if the result is not to be continuously half formed chaos. The principles of design laid out in the article by Keen and Gambino in the last *APL Users' Meeting Proceedings* [Ref. 2] have been very helpful in obtaining this. Principles, however, are of little use without a group of computer

professionals with the talent and integrity to follow them. We have been lucky to find such people in the staff of the local I.P. Sharp branch office.

Rather than continue to talk in abstractions, perhaps I should outline the case history of the system as it actually happened. In 1979, before development work began, budgeting in the province was quite fragmented. A batch mode computer system did exist, but its main function was to provide the figures for the published Estimates, which were displayed by Vote and Object of Expenditure. All the work needed to produce these very high level summaries was done manually, and making changes was a wretched business, as I can attest from personal experience.

Some four or five ministries were using Zero Base Budgeting (ZBB), which was actually the first form of program budgeting ever introduced on a wide scale into the province. Zero Base Budgeting had some very appealing features, and is in fact still in use in the province today, although in a highly modified form. Its chief advantage is that it permits the preplanning of balanced expenditure options by front line managers in the form of budget packages. If it is done well, front line managers can be guaranteed that they will have a budget they recognize at the end of the process. The overall funding level may not be what they would have liked, but it is at least one they have said they could live with. Too often governments cut budgets by cutting specific objects of expenditure, such as travel, and in the process produce a budget which is very difficult to manage. Also, the mechanism of ranking of budget packages has shown itself to be very useful for the ministry in sorting out its priorities.

However, in 1979, the ministries using ZBB were complaining about the clerical burden it presented. To oversimplify slightly, if each manager produced three budget packages, three times as many numbers had to be dealt with. This was a heavy burden, particularly in the more decentralized ministries. All ZBB was done manually.

The linkage between the ZBB ranked budget packages and the summary system's votes was rather vague, as was the linkage between the votes and the activity and responsibility centre coding scheme which was and is universally used to classify actual expenditures. More clerical effort was involved in making these linkages work, and often quite difficult work at that. How does one allocate a last minute ten percent cut among three hundred front line managers?

The first system which was developed, in November of 1979, simply made it possible to operate the vote summary system interactively. Part of the problem with the earlier system had been the three day processing time for any changes, and the six day processing time if a mistake was made the first time. Eliminating this turnaround time was a tangible and readily comprehensible benefit, and although this first system was never actually used, it opened the door for the next, rather more complex, system.

This second budgeting system, completed in April of 1980, incorporated the activity and responsibility centre chart of accounts used to record actual expenditures, so completing that linkage, but did not have any ZBB features. Total development cost to that point was some spare time of one person (myself) and a few hundred dollars in computer time.

This record system was impressive enough that it was possible to persuade two ministries, Forests and Lands, Parks and Housing, to use a system based on it for their 1980 budget process. This meant a further outlay of approximately \$15,000 but it also meant that a prototype which was refined in actual use could be created. The Lands, Parks and Housing system introduced all of the features needed to create and analyse

a classical Zero Base Budget, and it became the direct parent of the system which is now used in all ministries.

These two ministries were very enthusiastic users, and along with the Ministry of Finance, which joined in using the system later in the year, were a sufficient force to convince the authorities that cross government implementation was justifiable in 1981. This next phase in development was the most worrisome to date. Most of the people who would be called upon to operate what was becoming a rather complex computer system had never even touched a terminal before! Fortunately, it proved that there was ample capacity available to cope. Three things helped with this:

- The system had been developed with heavy input from actual users.
- A thorough user's manual was available.
- Assistance with problems was always available, even at odd hours of the night.

Also, the capacity to innovate to meet newly defined needs was often brought into play, although a rule was made that all changes must be augmentations, and not affect the basic operation of the system.

Since the system was in use in all ministries, it was natural to begin thinking about a system which would summarize all output for the use of top level decision makers, in other words the politicians. Accordingly, in late 1981, the first macro or cross-government summary system was devised. Because politicians traditionally looked at votes and objects of expenditure, that is how the system was organized. This would also provide the material for the published Estimates.

Unfortunately, there were a number of things wrong with this approach. The vote structure did not sufficiently concentrate attention on the issues that required decisions. Ministries, all by now using ZBB, wanted to discuss ranking, and groupings of budget packages which fell outside the pre-defined spending target, called issue papers. But the revenue picture was deteriorating rapidly enough that the original targets, set in May of 1981, were sadly out of date in February of 1982. Also, it was found that politicians could not easily use the ranking concept to make their decisions.

These problems led to the creation, in three days, of an issue paper system. This system eliminated all detail except a breakdown by vote of the expenditures within each ministry target, and a breakdown by issue paper of all expenditures from target up to the ministry's total funding request. This system was used extensively to keep score during the difficult set of meetings during which the Cabinet struck the 1982/83 expenditure budget.

This brings us almost up to the present. Most of the criteria for a good budgeting system I originally outlined have now been reasonably well met. Although the system is complex (there are other subsystems I have not described), it does produce an accurate, trustworthy budget. It was quite economical, considering the amount of processing it performed in the previous budget cycle, and promises to be much more so in the coming cycle. A new version of the ministry budget preparation system has just been released, and processing costs have been cut with it by as much as 60%. Our goal is to get computer costs down below what they were with the original batch mode summary system, and this seems quite feasible. Development cost is still less than

\$40,000 in total. There has been a considerable saving in the overtime bill as well. The effect of the "budget crunch" on accounts staff has been softened. Perhaps partially for this reason, and also probably because of the responsiveness of the system, user satisfaction at all levels from senior executive to terminal operator has been most gratifying.

The linkage to the actuals reporting system has now been developed to a point where all the work needed to prepare for a new budget year can be done by the computer with the help of a small table of monthly distribution patterns for cash flow monitoring.

Analytical capability is now very significant in each individual system, as is the capacity to react logically to change in a very short time. Many managers were given perspectives on the budget which have simply been unavailable up to this point in time, and this is having an impact on the whole process. For example, the system has shown that there is little utility in cutting by object of expenditure. Many simply are not large enough to make a real difference to the overall budget and their elimination makes a budget unworkable in practice. It is in achieving these higher aims that the system still has some distance to go however, as I will discuss in the next section of this paper.

Future directions

It now appears that a point has been reached in the development of the system that it is beginning to influence its environment as much as its environment influences it. The capacity to do analysis accurately and quickly, and to make changes easily, has caused people using the system to revise their ideas of what is possible, and therefore to demand more.

The major flaw in the system last year was that the direct linkage between the budget package prepared by the front line manager and the issue paper considered in Cabinet was lost. Consequently, the manager's input, and even the input of the ministry as a whole, was in danger of being lost when Cabinet was making final decisions. In this year's version of the system this direct link is in place, and in such a way that everyone involved in the process should feel that they have had their just say in decisions.

In this way it is to be hoped that a budget which is perceived as less arbitrary by those who will have to live with it will be created. It also may be hoped that the amount of data available, and its better organization, will make it easier to make decisions throughout the budget process which in turn will mean more value for money for the taxpayer.

Another benefit which this system has given the government is that it has demonstrated very clearly that the adaptive design approach works. It is probably about the most successful system the government has ever had, yet nearly all the rules of traditional system development were broken by it. As technology changes, the future for this approach is bright. This is as it should be. After all, it follows the design pattern for the universe. The traditional approach contradicts it in every way.

References

1. Gall, John. *General Systemantics*. Time Books (Harper Row), 1977.
2. Keen, Peter G.W., and Thomas T. Gambino. "The Mythical Man Month Revisited: Building a Decision Support System in APL." *1980 APL Users Meeting Proceedings*, pp. 630-648.

Acknowledgements

I would like to gratefully acknowledge the support and expertise provided by Mike Powell and the rest of the staff of I.P. Sharp's Victoria branch, without whom this system could never have been developed.

CURRENT SIGAPL ACTIVITIES

Ray P. Polivka
Chairman
SIGAPL

SIGAPL, the Special Interest Group on APL, is the name for an international organization of people who are interested in APL, its use, development, and promotion.

The name SIGAPL derives from the fact that this organization is one of several special interest groups within the ACM, the society for the computing community. SIGAPL has grown with the growth of APL. The first formal APL organization was the APL committee of the Interactive System Project within the SHARE organization, established in 1969. The prime mover in this early stage was Garth Foster of Syracuse University. He undertook to produce an APL newsletter. He acted as its editor, distributor, and publisher for several years. This newsletter, which assumed the name of *Quote Quad*, has become the focal point of communication and news among the APL community.

An informal organization grew up around *Quote Quad*. By 1972, however, the community had grown so large that managing it became too much of a burden for a few volunteers. Thus in 1972-73 the APL group affiliated with the ACM, becoming known as STAPL, the Special Technical Committee on APL. STAPL entered the ACM as part of the Special Interest Group on Programming Languages (SIGPLAN), since from ACM's point of view APL was just another programming language. In this structure, the ACM was able to provide the administrative assistance that an organization needs: printing and mailing newsletters, running elections, collecting dues, safeguarding funds, and enforcing fiscal responsibility.

But, STAPL continued to grow. It became larger and more active than many of the other SIGs. It began to have a budgetary and identity problem within SIGPLAN. Thus when the then current chairman, Gene McDonnell, requested that STAPL become a full-fledged independent SIG, the ACM approved the change. STAPL became SIGAPL.

SIGAPL Today

SIGAPL does many things for its members. Foremost among its activities is the publication of *Quote Quad*, a quarterly publication containing articles, algorithms, announcements, book reviews, problems, correspondence, and other material of interest to the general APL community.

Beside publishing *Quote Quad*, SIGAPL, sponsors international conferences on APL, an extremely valuable means of communicating and exchanging information. SIGAPL has sponsored or cooperated with most of the large international APL conferences. These meetings have been held more or less annually since 1969. The places were Binghamton NY (1969), Greenbelt MD (1970), Paris France (1971), Atlanta GA (1972), Toronto Canada (1973), Anaheim CA (1974), Pisa Italy (1975), Ottawa Canada (1976), Rochester NY (1979), Noordwijkerhout Netherlands (1980), San Francisco CA (1981), and Heidelberg Germany (1982).

In addition to full scale conferences, SIGAPL sponsors specialized workshops. These are usually small meetings of from 10-60 people. At these workshops, specialists in a particular area of APL can exchange their ideas and concerns. The most recent workshop was held in New York City, and dealt with numerical methods and APL.

At the Rochester conference in 1979, an IBM APL standard document was presented. From that document grew the effort to produce a general APL standard. The effort to achieve a standard is an international one which is a complicated and slow process. It is continuing today and SIGAPL is supporting this effort.

Today SIGAPL is an international organization with 1659 members. Almost 20% of our membership is from outside of North America. SIGAPL strives to serve any individual interested in APL regardless of country boundaries.

Future directions and plans

As SIGAPL looks to the future, it finds many things that could be done, many things that are waiting to be done, and many things that need to be done. Foremost in any future plans is the promotion and strengthening of *Quote Quad*. Essentially this is a two part endeavor. The editor has to ensure the worthiness of the material and the timeliness of its publication. The second part involves the APL community. They must provide the content of *Quote Quad*, the articles, correspondence, algorithms, problems, news items, puzzles, and recreational items. No editor can produce a document unless he has something to put into it. Let me invite you to submit material; we need your participation. To help the editor, SIGAPL has placed into its budget a small amount to purchase some equipment or services which would help produce the publication more rapidly. I am sure that the editor would be interested in the community's opinions. Another problem which *Quote Quad* faces today is the increasing number of APL publications springing up. While we encourage an author to submit material to other journals, we do need to support *Quote Quad*. It is still the document to use to reach the general APL community. SIGAPL plans to continue its policy of publishing as an issue of *Quote Quad* the proceedings of any APL conference that SIGAPL sponsors or participates cooperatively in. This is done to assure the widest distribution of APL material.

SIGAPL plans to sponsor future APL conferences in North America and cooperate with those planning conferences outside of North America. The next SIGAPL sponsored conference is APL 83 to be held in Washington DC April, 1983. SIGAPL is also cooperating with Finn APL on the APL 84 conference to be held in Helsinki, Finland. In addition, as each country and local chapter establishes its own APL group, we hope to maintain close contact with their organizations, publishing news and exchanging information. Some of the APL organizations that exist outside North America are UK APL in London, The APL Club of Germany, Finn APL in Helsinki, Swiss

APL User Group, The Werkgroep APL in the Netherlands, and the Belgian APL-CAM Users Society.

As concentrations of APL users become established within geographic regions, local SIG chapters are possible. Today several local chapters are in some phase of formation. The New York City local chapter has a tentative set of bylaws and has been meeting over the past two years. Seattle has been holding informal meetings for about a year; they are moving toward a more formal structure. Discussions about the formation of local chapters in Los Angeles, Rochester NY, Chicago, and Atlanta have occurred. National SIGAPL wishes very much to encourage local SIGAPL chapters. Local chapters with their APL enthusiasts can do a great deal more than national SIGAPL to promote APL.

With a strong percentage of overseas members, it seems that they should be represented on the SIGAPL board. This person could serve as the communication link between these members and SIGAPL. This board member could also be the focal point for coordination between SIGAPL and any national organization in such items as conference planning.

Today more than ever, it is timely for SIGAPL to bring the language more to the attention of the general computing community. Presenting APL tutorials or workshops at the general ACM and NCC conferences are two ways in which this can happen.

Reaching the new and future computer scientist is important. Establishing a student paper contest could be a way of attracting student interest. An APL oriented summer camp is another possibility. The presentation of APL at teacher conferences will reach another important audience. Creation of APL literature aimed directly at the younger audience is also needed.

This is an exciting time to be involved in APL. There are many things that can be done which will benefit the individual APL user and the community as a whole; *but*—warning needs to be issued. All of the activities that are happening or being contemplated are by volunteers, by people who are doing it out of devotion, joy, and dedication. Regardless of the enthusiasm of the SIGAPL board, the only way any particular thing is going to happen is if someone takes it upon himself to pick it up and make it his own. We constantly need people who will step forward and volunteer to serve.

There are many ways in which this can happen:

- Establish and participate in local chapters
- Submit material to *Quote Quad*
- Promote and host specialized workshops
- Submit papers reflecting your work to APL conferences and to non-APL conferences
- Attend the APL conferences
- Convince your employer to support your participation in SIGAPL's professional activities

- Talk to others about APL and your use of it
- Talk to any of the SIGAPL board members about your ideas and offer to follow up on them
- Join us (SIGAPL)

Once again, the objective of SIGAPL is to advance the language APL in all its aspects. The achievement of that objective depends upon the participation and cooperation of all interested individuals. If you can identify with this objective, join us and actively participate.

MOORE CORPORATION'S FINANCIAL CONSOLIDATION SYSTEM

Thomas P. Courtney
Assistant Controller
and
Paul Maclauchlan
Specialist, APL Development
Moore Corporation Limited
Toronto, Ontario

Abstract

Moore Corporation Limited is a multinational organization based in Canada, best known over the years as the largest manufacturer of business forms in the world. Moore operates 131 manufacturing plants in 39 countries. Total annual revenue is approximately U.S. \$1.9 billion. Operations of this magnitude and breadth are served best by a computerized financial reporting, currency translation and consolidation system. In 1980 Moore evaluated several such packages. After extensive examination, Moore decided to adopt I.P. Sharp's CONSOL package.

To date, Moore has implemented several CONSOL based systems as part of its plan to install a worldwide management reporting and financial consolidation system. While well underway, the system will not be completed for some time.

Currently, data is entered into the system at the Corporate Head Office in Toronto, and at two remote subconsolidation locations which are primarily responsible for International Operations.

The authors will discuss the evolution of the system and the plans for data to be entered directly by each subsidiary in its own currency so as to produce consolidated reports in the global U.S. currency.

Introduction and history

Moore initially perceived the need for the eventual computerization of its financial reporting soon after it acquired 100% control of Lamson Industries, a U.K. public company, in 1977. Up until that time Moore had been able to consolidate its large but relatively few operating units in North America on a manual basis. The advent of Lamson, with its many subsidiaries spread throughout the U.K., Europe, Africa and Australasia, presented a new challenge involving logistical and more complex accounting problems.

As a first step, it was decided a new accounting manual had to be written to replace the existing manuals in Moore and the former Lamson organization. The purpose of this manual was to establish a standard, uniform manner of reporting financial data,

including financial statement formats, reporting deadlines, etc. The new Moore statement formats were designed to include line codes, format references, etc., in order to permit the transmission at some future date of coded key operating results and budget information.

The complete adoption and implementation of the Moore Accounting Manual on a worldwide basis was a prerequisite to computerization and this process took some time to accomplish, as is often the case following the conversion to a new system of reporting.

Following the introduction of the Accounting Manual at the beginning of 1979, a feasibility study commenced with the object of developing a common worldwide communications, financial reporting, and consolidation system.

During the course of 1979 several options were evaluated and resulted in the choice of I.P. Sharp as the organization best suited to meet Moore's needs for the following reasons:

- Well represented internationally—in almost all of the jurisdictions in which Moore operates
- High degree of reliability of the I.P. Sharp communications network
- Close proximity to Moore's Corporate Head Office in Toronto
- I.P. Sharp's experience and reputation in developing such systems as confirmed by Moore personnel's visits to other multinational clients of I.P. Sharp

In light of the above, Moore selected I.P. Sharp to assist in the development and implementation of a financial reporting and consolidation system.

By way of background, it should be mentioned that the name "Moore" was introduced throughout the old Lamson organization, and the international headquarters group based in London, England became known as Moore International Division, London (MIDL). The MIDL staff remained responsible for the subconsolidation of financial data in respect of all the operating units in its geographic area of responsibility, essentially comprising the U.K., Europe, Africa, and Australasia. It had used a time-sharing service for some two years prior to Corporate Head Office (CHO) becoming involved in computerization and had reached the stage of producing a computerized financial consolidation for International Operations.

It was thus considered important that personnel at CHO maintain continuous liaison with MIDL in the definition and development of the proposed system, in order to avoid duplication of effort. A first step involved MIDL moving its computer applications from its previous timesharing service onto I.P. Sharp.

The computer system as envisaged at the outset by CHO was for financial data to be entered into the I.P. Sharp system at the reporting unit level, validated and checked, translated, and consolidated. Such a system would allow for aggregations and reports at various levels of responsibility within the overall consolidation, including consolidations along legal entity lines and by paths in accordance with areas of management responsibility.

The financial consolidation system was developed in Toronto using CONSOL, the

applications software developed by I.P. Sharp. Initially, two CONSOL systems were used for the financial consolidation, producing consolidated balance sheets and income statements on a quarterly time frame. In these early stages the computer was being used as a large calculating machine since the numbers were entered into the system and aggregated by consolidation path in U.S. dollars only. An interface program was written to obtain the numbers in respect of the MIDL group, since these were part of a computer system having a different database file structure to CONSOL. Further development of the CHO system in terms of the introduction of a foreign currency translation feature was delayed pending the determination by the Canadian Institute of Chartered Accountants (CICA) of a definitive accounting standard. However, the completion of a translation standard by the CICA became a lengthy process because of the complexity of the issues involved. It was at this point that Moore decided that it would move ahead with a translation routine following the present currency conversion rules, in the knowledge that at some future date the rules would change.

In the meantime, considerable experience and benefits had been gained from using the basic CONSOL system, although the goal remained of adopting one computer system for the whole organization. The CHO CONSOL system had been built up differently from the MIDL system in that it was developed to be more user-friendly. After a very short period of training, staff in the consolidation team were able to input data and pull off reports. A user manual was also introduced to act as a permanent source of reference. The MIDL approach had been to design their system for sophisticated users, involving a great deal of person-computer interaction during the translation/consolidation process. This created the risk of inadequate cover in the event of staff sickness or terminations.

However, as CHO experience grew it enabled management to have confidence in expanding the system on a worldwide basis, with the intention that all units would input directly into the I.P. Sharp system. The decision to proceed along these lines was confirmed earlier this year.

It was decided that any MIDL software that could be used would form part of the new system. Among other things, this would include using their sub-system for fixed asset translation, the movements on which are required to be stored separately for each time period, a system for the elimination of intercompany balances, tax provision reconciliation and analysis, and currency translation variance analysis.

At the present time all the efforts of CHO and MIDL are directed towards the establishment of a single, worldwide consolidation system. The consolidation itself will be performed at CHO, and the resulting information and reports will be made available to the various Moore management groups. The timetable for this project has an established deadline of Spring 1983.

Implementation problems and solutions

There have been three main areas in which Moore has faced development problems, and found workable solutions. Each will be covered in turn. They are:

- The handling of consolidation and other head office adjustments
- The development of a uniform translation procedure within CONSOL
- A common input procedure for operating units to follow

Adjustments

In any consolidation, adjusting entries are required. At Moore there are basically three types:

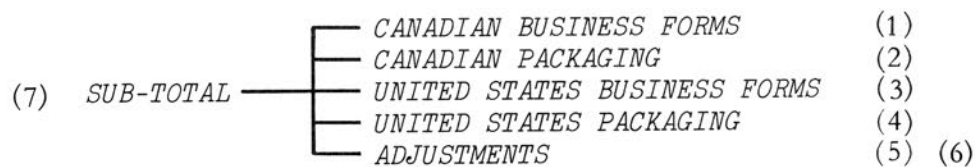
- Consolidation adjustments—elimination of intercompany sales, receivables/ payables and investments
- Corrections—of original data received from reporting units
- Head Office adjustments—made in the latter stages of the consolidation

These adjustments created concern when the consolidation system was being designed since there was no facility within CONSOL for recording them. Some of the options considered at the time included: recording the adjustments directly to the nodes affected, creating a distinct node for each adjustment, creating a single node for the adjustment, etc., etc.

The final solution was to create nodes which would hold only adjustments. This was later refined and will be discussed later. The manual worksheets for our consolidation had headings as follows (examples only):

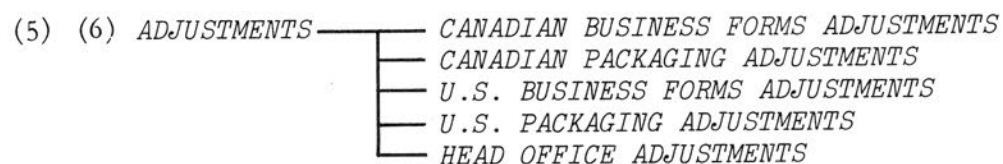
(1)	(2)	(3)	(4)	(5)	(6)	(7)
Canadian Business Forms	Canadian Packaging Products	United States Business Forms	United States Packaging Products	Adjustments		Sub-Total
				DR	CR	

In order to represent this convention in CONSOL, the consolidation tree was set up as follows:

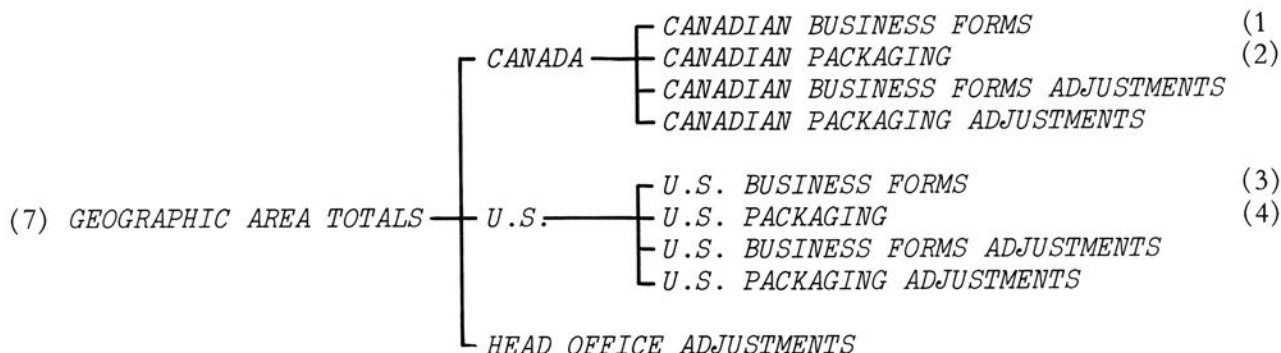


Thus, any adjustments that would appear in columns (5) or (6) on the existing manual form would appear in the *ADJUSTMENTS* node in the computer.

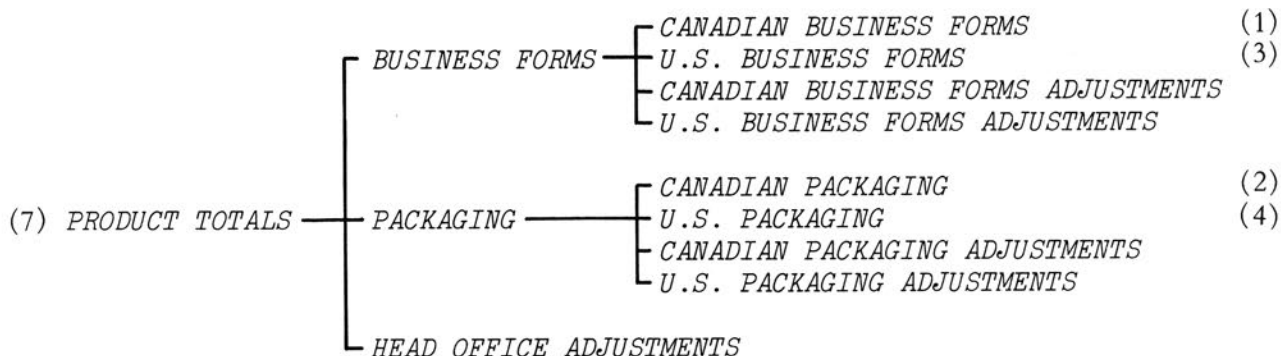
The above method worked fine for the initial “legal” consolidation, but soon more sophisticated analysis was required. The analysis desired was a segregated operating result by product and by geographic area. In order to do this, additional paths were created, but the adjustments were still lumped together in one node. They had to be segregated as follows:



The last node on this list was for any adjustments that were not segregated. New trees were then created, to produce consolidated statements on a segregated basis:



and:



This manner of handling adjustments has provided a good framework within which to work. All entries are now classified and recorded according to geographic and product segment, and the resulting analysis can be obtained concurrent with the legal consolidation.

Translation

The first step in development was to assemble the Moore translation rules into a definitive written procedure. The rules had to be refined to a degree where all possibilities were covered. Then this procedure was converted to a semi-English/semi-APL series of formulas. A meeting was then arranged with Moore's I.P. Sharp representatives and Margaret Reilly, author of CONSOL, to discuss implementing the detailed rules within CONSOL.

The Moore rules for translation were deemed too complex to be adapted to the conventional CONSOL conversion routines. The existing information contained in Moore's current computer system was not sufficient to cover the requirements of the translation procedure. In addition, the cost of doing multiple conversions (i.e., for every report and consolidation) was too high. The final design comprised three CONSOL application systems:

- LOCCON to hold the local currency results
- FINCON (our existing system) to hold U.S. dollar results
- XCON to hold the additional information not contained in the present systems

As mentioned earlier, the fixed asset translation information would come from a stand alone sub-system, already in use by the MIDL group.

Local currency amounts on the financial statements are translated to U.S. dollars at various exchange rates. These include: period closing rates, period average rates, and historical rates. Items translated at "standard" rates (period closing or period average) have the exchange rates for the current period applied in a straightforward manner, one which would suit CONSOL easily. The difficulty came in dealing with the remaining "non-standard" (historical items).

An example may prove useful:

Balance Sheet (Local Currency)	<u>Prior Period</u>	<u>Current Period</u>
Cash	100	150
Investment In Subsidiary	2,000	1,500

The cash, a "standard" item, is translated at the exchange rate on the day the statements are prepared. If the closing rate for the prior period was .85 (\$ per unit of local currency) and is now .80, the cash will be translated as follows:

Balance Sheet	<u>Prior Period</u>			<u>Current Period</u>		
	<u>LC</u>	<u>Rate</u>	<u>\$</u>	<u>LC</u>	<u>Rate</u>	<u>\$</u>
Cash	100	.85	85	150	.80	120
Investment In Subsidiary	2,000			1,500		

The translation of the amount of the investment in subsidiary, a "non standard" item, is not as straightforward. The U.S. dollar amount must reflect the value of the investment on the date it was made. To record this, a schedule must be built up showing the component investments which support the balance sheet amount. The historical rate schedule could look as follows (at the end of the prior period):

	<u>LC</u>	<u>Rate</u>	<u>\$</u>
Subsidiary A	1,000	1.00	1,000
Subsidiary B	500	.98	490
Subsidiary C	<u>500</u>	.90	<u>450</u>
	2,000		1,940
	=====		=====

Since the preparation of the last balance sheet, there has been a sale of subsidiary C. The historical rate schedule for the current period shows:

	<u>LC</u>	<u>Rate</u>	<u>\$</u>
Subsidiary A	1,000	1.00	1,000
Subsidiary B	500	.98	490
	1,500		1,490
	=====		=====

The sale or acquisition of a subsidiary is an uncommon occurrence. For this reason, the translation of this item is handled "off-line" from the main consolidation. The

XCON system is set up to hold the various local currency and U.S. equivalent amounts for the "non-standard" rate items. The CONSOL form will look like this (for the current period):

ITEM	ITEM NAME	CURRENT PERIOD
XX1	LOB-LOCAL CURRENCY OPENING BALANCE	2,000
XX2	LAD-LOCAL CURRENCY ADDITIONS	-
XX3	LD-LOCAL CURRENCY DISPOSALS	500
XX4	LCB-LOCAL CURRENCY CLOSING BALANCE	1,500
XX5	UOB-U.S. DOLLAR OPENING BALANCE	1,940
XX6	UA-U.S. DOLLAR ADDITIONS	-
XX7	UD-U.S. DOLLAR DISPOSALS	450
XX8	UCB-U.S. DOLLAR CLOSING BALANCE	1,490

Source: XX1 = Prior Period Item XX4
 XX5 = Prior Period Item XX8
 XX2, XX3 = Input From Company Financial Statements
 XX6, XX7 = Calculated "Off-Line"
 XX4, XX8 = Calculated Within CONSOL

Once the data is entered into XCON, and calculated, the translation is run. The translate routine applies the standard rates to the standard rate items (i.e., Cash in our example) and then transfers the U.S. dollar amounts, for the non-standard items, from XCON into FINCON, the U.S. dollar database. The resulting balance sheet, in our example, would show:

Balance Sheet	Prior Period			Current Period		
	LC	Rate	\$	LC	Rate	\$
Cash	100	.85	85	150	.80	120
Investment In Subsidiary	2,000	XCON	1,940	1,500	XCON	1,490

The Moore approach is that once the required data has been input by the user, the translation should proceed with no manual intervention. This approach ensures a clear audit trail which is a desired byproduct where there are as many calculations as in the Moore translation/consolidation system.

Direct input

A great deal of time was being spent each quarter-end and each month by Moore's Head Office staff transferring data from each company's financial statements into the database. The consolidation group in London, England (MIDL) spearheaded development in this area.

The first problem was hardware. The installation of an I.P. Sharp compatible terminal at each operating location could not be justified on economic grounds because of the anticipated low utilization; i.e., only one day a month. The solution came when MIDL discovered that I.P. Sharp supports telex input at certain nodes. All of the MIDL locations had a telex machine, all they needed were instructions and an account number.

MIDL developed a software package which the companies use to transmit their results, via telex, into a holding file on the I.P. Sharp system. The holding file is used as a

buffer between the reporting units and the databases, in order that the integrity of the main database would be maintained.

The direct input system has evolved since that time and is now on the threshold of being introduced worldwide in Moore. The system supports the input of all the major financial statements and is being expanded to include numerous support schedules. In addition, there is an editing feature that ensures arithmetic integrity and agreement between forms, and highlights period to period variances on which the reporting units are required to comment.

Perhaps the most valuable feature of this system is the facility for balancing intercompany receivables and payables. A reporting unit records its intercompany balances (ICB's) by company and this information is matched to the sister company's input. Any differences are highlighted and must be reconciled. This feature has saved hours of time at the Head Office since most out-of-balance situations are corrected without the involvement of the consolidation group.

Conclusion

Moore's translation/consolidation system is constantly evolving and, at times, seems to have a life of its own. We are moving quickly towards our target of an integrated information system, which will support all of Moore's financial reporting requirements. The use of I.P. Sharp's CONSOL package has provided a solid base to work from and allowed us to implement computer consolidations in a short period of time. If we had to start from scratch without a software package, we could not have progressed as far this quickly.

TRACKING AND MEASURING THE FORECAST

Michael J. Tenalio
Production Administrator
Systems Division
Sybron/Taylor Instrument Company
Rochester, New York

Being faced with recessionary times has enhanced the need for manufacturing efficiency. This "efficient" trend encompasses a wide array of cost cutting methods that include and incorporate labor, hardware, and other valuable resources. Just as important is the need for operational efficiency in the areas of sales forecasting, capacity requirements planning, and material requirements planning.

It is the latter concentration that will be approached here simply because forecasting future demand with reasonable accuracy is essential when you take part in a competitive business that requires purchasing and manufacturing of long lead time items.

BACKGROUND

The Taylor Instrument division of Sybron is involved in the design and manufacture of sophisticated, computerized process control systems for industrial applications.

Manufacture and assembly of the hardware needed for these systems is performed at different sites.

Manufacturing produces parts based on a forecast by the Marketing department. The final Assembly, Test and Inspection plant orders parts from Manufacturing as required and inventory is kept at the manufacturing site.

THE PROBLEM

Marketing's forecast was based on the number of systems they could sell. Translating this into parts requirements was, historically, based on "typical" definitions. (A typical definition or parts requirements list exists for each of the four basic product lines that Taylor produces.) Unfortunately, due to the customized nature of most of the orders for systems, parts requirements often differ from the "typical" definitions.

Forecasting of parts, as opposed to systems, was not completely accurate and consequently inventory levels were sub-optimal.

Forecasting was further complicated by a combination of factors:

- A growing demand for updates to systems
- Maintenance of customer systems
- Changes in delivery dates
- Maintenance of internal systems

It was increasingly evident that an effective means of communication between Marketing, Manufacturing and Assembly was needed to:

- ensure prompt deliveries and thus avoid customer dissatisfaction
- optimize inventory levels

SYSTEM DESIGN

Our original goal was to design an effective parts-inventory tracking and forecasting system that would provide the following:

- A month by month forecast of parts requirements extending 18 months into the future. This forecast had to be acceptable to Assembly, Manufacturing, and Marketing.
- A means of comparing the forecast with actual parts used.
- The ability to list customer/parts information by month or by customer.
- The ability to adjust the forecast to reflect incoming orders and deadline changes, and generate an order list based on these changes.

The Systems Unit Order Budget system (SUOB) was a first pass at this problem and the Parts Unit Order Budget system (PUOB) was a necessary supplement to manage the maintenance and update parts requirements.

These systems perform basically the same functions, the major difference being how the forecast data is stored. The SUOB retains a systems forecast which is expanded at report time using typical definitions, whereas the PUOB retains a forecast by part.

Because the PUOB system was designed with the benefit of previous experience with SUOB, and to avoid confusing the reader, the remainder of this paper will be devoted to PUOB.

SYSTEM DESCRIPTION

Parts requirements for maintenance and updating systems are divided into two product categories, MOD3 and COMPUTER. A parts list is maintained for both MOD3 and COMPUTERS. These lists were entered once at system start up, but a facility exists to add new parts and delete any that may become obsolescent.

Parts Unit Order Budget System

An associated 30 month forecast, by part, is stored on file—12 months historical and

18 months future. Entering the 18 month forecast is done in two steps and allows for entry of a future 12 month forecast as early as July of preceding year.

The forecast is automatically updated each month. The oldest forecast month is discarded and an extra slot for a forecast is added (18 months from current month).

We have the option of distributing the forecast over 12 months or 4 quarters—(where forecast figures are not divisible by 4 or 12, the residue is spread evenly over the early months/quarters).

New forecasts are entered just once a year, in July.

The ability to make changes throughout the year to either future or historical forecast data is provided for.

Project data associated with customers is entered as it becomes available. Sometimes this is day by day, sometimes weekly.

Each project record comprises the following:

- The customer name
- Whether the order is firm or allocated [firm being contractual orders and allocated being, potential orders]
- The date when the project is to be started.
- The type: COMPUTER or MOD3
- The detail: a list of the number of parts required from the appropriate parts list. Detail data is stored in matrices, one for each of the 30 months in the variable *MONTHS*. Rows correspond to parts, columns correspond to customers.
- The position: a final field gives the position of that particular customer's detail in the matrix for that month.

REPORTS

The PUOB report is run monthly. Other reports are run as they are needed any time during the month. Such occasional reports include:

- Project Review
- Accuracy Report
- Detailed Project listing by month
- Detailed Project listing by customer
- Parts listings
- Forecast listings
- Customer listing

PUOB report

The PUOB report is run twice a month—once for MOD3 and once for COMPUTERS.

The report generates a monthly parts needs list for the current 6 quarters and a summary report showing quarterly totals with a grand total.

This report displays, by part, the Firm and Allocated orders, the forecast and the leftover parts, called "FILL". The calculation used to arrive at FILL is:

$$\text{FILL} = \text{FORECAST} - (\text{ALLOCATED ORDERS} + \text{FIRM ORDERS}).$$

A special enhancement affecting the PUOB report is:

Inventory carry forward: You have the option of carrying inventory forward for specified months. If this option is used any fill is carried forward and used to reduce the forecast in succeeding month(s).

N.B. Negative fill is shown in the month of occurrence and is not carried forward because the PUOB report is used as an "order" for parts.

This is the key report in the system, it shows at a glance the time frames where a potential parts shortage or overage can occur. Remedial steps can be taken early and appropriate inventory levels are maintained.

PART UNIT ORDER BUDGET ALL COMPUTER LINES																
PART	JULY 82				AUGUST 82				SEPTEMBER82				3RD QTR82			
	FIRM	ALLC	FILL	TOT	FIRM	ALLC	FILL	TOT	FIRM	ALLC	FILL	TOT	FIRM	ALLC	FILL	TOT
23E4566	1	0	0	1	10	0	2	12	4							
101R34	3	0	0	3	0	0	4	4	5	0	8	12	15	0	10	25
123S23	0	0	0	0	3	0	1	4	8	0	-1	4	8	0	-3	11
178D23	64	0	3	67	1	0	2	3	2	0	-4	4	11	0	-3	8
345T67	94	0	0	94	1	0	1	2	2	0	1	3	67	0	6	73
SK2345	0	0	0	0	0	0	1	1	0	0	0	2	97	0	1	98
4830NA01100	6	0	0	6	12	0	1	13	13	0	0	0	0	0	1	1
4832NAX18	0	0	0	0	2	0	2	4	5	0	0	13	31	0	1	32
4836NA01100	3	0	0	3	2	0	2	4	4	0	-1	4	7	0	1	8
4848R	0	0	0	0	0	0	1	1	1	0	0	4	9	0	2	11
4855N	72	0	1	73	1	0	2	3	10	0	-1	0	1	0	0	1
4858NA21000	84	0	0	84	1	0	1	2	3	0	-7	3	83	0	-4	79
4861N	0	0	0	0	0	0	1	1	0	0	-1	2	88	0	0	88
4862N	1	0	2	3	16	0	1	17	8	0	0	0	0	0	1	1
4863N	1	0	0	1	12	0	3	15	4	0	9	17	25	0	12	37
4870N	3	0	0	3	2	0	2	4	5	0	11	15	17	0	14	31
										0	-1	4	10	0	1	11

Project review

The Project Review report displays all the projects on file or all those in selected months.

The report is sorted by month and lists customer name, date, product type (MOD3 or COMPUTER) and whether Firm or Allocated.

PARTS UNIT ORDER BUDGET			
PROJECT REVIEW			
<u>MONTH</u>	<u>CUSTOMER</u>	<u>TYPE</u>	<u>FIRM</u>
AUG 82	CUSTOMER 30	COMP	F
SEP 82	CUSTOMER 10	COMP	F
	CUSTOMER 12	COMP	F
	CUSTOMER 14	COMP	F
	CUSTOMER 28	MOD3	F

Accuracy

The Accuracy report compares the actual usage of parts over a period of time with the Forecast. The report shows the Average usage, the variance and the % of variance:

$$\text{Average} = \frac{\text{FORECAST} - \text{FILL}}{\text{FORECAST}} \times 100$$

$$\text{Variance} = -\text{FILL}$$

$$\% \text{ Variance} = \frac{-\text{FILL}}{\text{FORECAST}} \times 100$$

FORECAST COMPUTER COMPARED TO 3 MONTH AVERAGE FROM 7/82 TO 9/82 INCLUSIVE					
<u>REF</u>	<u>PART</u>	<u>FCST</u> <u>QTY</u>	<u>AVG</u>	<u>VARIANCE</u>	
				<u>QTY</u>	<u>PCT</u>
1	23E4566	25	60.0	-10	-40.0
2	101R34	11	72.7	-3	-27.3
3	123S23	8	137.5	3	37.5
4	178D23	73	91.8	-6	-8.2
5	345T67	98	99.0	-1	-1.0
6	SK2345	1		-1	-100.0
7	4830NA01100	32	96.9	-1	-3.1
8	4832NAX18	8	87.5	-1	-12.5
9	4836NA01100	11	81.8	-2	-18.2
10	4848R	1	100.0		
11	4855N	79	105.1	4	5.1
12	4858NA21000	88	100.0		
13	4861N	1		-1	-100.0
14	4862N	37	67.6	-12	-32.4
15	4863N	31	54.8	-14	-45.2
16	4870N	11	90.9	-1	-9.1

← Usage in this case was 15.

Detailed project listing

Detailed data associated with projects can be accessed by month or by project number. A maximum of 15 projects can be displayed on one report.

PARTS ORDER BUDGET REPORT				
COMPUTER				
<u>INDEX</u>	<u>CUSTOMER</u>	<u>FIRM</u> <u>ALLOC</u>	<u>MONTH</u>	
10	CUSTOMER 10	F	9/82	
12	CUSTOMER 12	F	9/82	
14	CUSTOMER 14	F	9/82	
CUSTOMER INDICES				
<u>REF.</u>	<u>PART</u>	<u>10</u>	<u>12</u>	<u>14</u>
1.	23E4566	1	1	2
2.	101R34	2	2	1
3.	123S23	0	0	8
4.	178D23	1	1	0
5.	345T67	1	1	0
6.	SK2345	0	0	0
7.	4830NA01100	5	6	2
8.	4832NAX18	1	1	3
9.	4836NA01100	1	1	2
10.	4848R	0	0	1
11.	4855N	1	1	8
12.	4858NA21000	1	2	0
13.	4861N	0	0	0
14.	4862N	3	5	0
15.	4863N	1	1	2
16.	4870N	1	1	3

Parts listing

The parts on file can be listed on one page together with an associated reference number.

COMPUTER PARTS LIST	
SEPTEMBER 8, 1982	
<u>REF</u>	<u>PART</u>
1	23E4566
2	101R34
3	123S23
4	178D23
5	345T67
6	SK2345
7	4830NA01100
8	4832NAX18
9	4836NA01100
10	4848R
11	4855N
12	4858NA21000
13	4861N
14	4862N
15	4863N
16	4870N

Input sheet

The input sheet is simply a parts listing with rows at the top of the report to enter customer names and columns alongside the part numbers for entering quantity information.

COMP INPUT SHEET																	
SEPTEMBER 8, 1982																	
<u>CUSTOMER</u>	<u>CUST REF</u>	<u>ALLOC</u>	<u>(MM/YY)</u>	<u>PROJ ID</u>													
1. _____	_____	_____	_____	_____													
2. _____	_____	_____	_____	_____													
3. _____	_____	_____	_____	_____													
4. _____	_____	_____	_____	_____													
5. _____	_____	_____	_____	_____													
6. _____	_____	_____	_____	_____													
7. _____	_____	_____	_____	_____													
8. _____	_____	_____	_____	_____													
9. _____	_____	_____	_____	_____													
10. _____	_____	_____	_____	_____													
11. _____	_____	_____	_____	_____													
12. _____	_____	_____	_____	_____													
13. _____	_____	_____	_____	_____													
14. _____	_____	_____	_____	_____													
15. _____	_____	_____	_____	_____													

<u>REF</u>	<u>PART</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>	<u>7</u>	<u>8</u>	<u>9</u>	<u>10</u>	<u>11</u>	<u>12</u>	<u>13</u>	<u>14</u>	<u>15</u>	_____	_____	_____	_____
1.	23E4566																			
2.	101R34																			
3.	123S23																			
4.	178D23																			
5.	345T67																			
6.	SK2345																			
7.	4830NA01100																			
8.	4832NAX18																			
9.	4836NA01100																			
10.	4848R																			
11.	4855N																			
12.	4858NA21000																			
13.	4861N																			
14.	4862N																			
15.	4863N																			
16.	4870N																			

Forecast Listings

The forecast report is a listing of the number of parts forecast by month, for the period selected.

FORECAST FOR 6 MONTHS BEGINNING MAY							82
SEPTEMBER 8, 1982							
<u>REF</u>	<u>PART</u>	<u>8205</u>	<u>8206</u>	<u>8207</u>	<u>8208</u>	<u>8209</u>	<u>8210</u>
1.	23E4566	1	0	1	12	12	4
2.	101R34	1	0	3	4	4	5
3.	123S23	0	2	0	4	4	10
4.	178D23	18	0	67	3	3	2
5.	345T67	24	67	94	2	2	2
6.	SK2345	0	9	0	1	0	0
7.	4830NA01100	3	1	6	13	13	19
8.	4832NAX18	1	0	0	4	4	5
9.	4836NA01100	1	0	3	4	4	4
10.	4848R	0	2	0	1	0	0
11.	4855N	18	0	73	3	3	10
12.	4858NA21000	24	67	84	2	2	3
13.	4861N	0	9	0	1	0	0
14.	4862N	3	1	3	17	17	8
15.	4863N	1	0	1	15	15	4
16.	4870N	1	0	3	4	4	6

Customer listing

The customer listing is a list of customers in alphabetical order by product type. The report is used to generate the reference numbers needed to refer to projects when making changes to, or listing project data.

As customers are added they will be assigned the next reference number in sequence and therefore this listing should only be necessary when a listing has become obsolete because of:

- many additions
- deletions
- dropping off of 12 month old data via the *ROLLOVER* function.

<i>CUSTOMER LISTING</i>		
<u>REF.</u>	<u>CUSTOMER</u>	<u>PROD</u>
20	CUSTOMER 20	1
21	CUSTOMER 21	1
22	CUSTOMER 22	1
23	CUSTOMER 23	1
24	CUSTOMER 24	1
25	CUSTOMER 25	1
26	CUSTOMER 26	1
27	CUSTOMER 27	1
28	CUSTOMER 28	1
29	CUSTOMER 29	1
1	CUSTOMER 1	2
2	CUSTOMER 2	2
3	CUSTOMER 3	2
4	CUSTOMER 4	2
5	CUSTOMER 5	2
6	CUSTOMER 6	2
7	CUSTOMER 7	2
8	CUSTOMER 8	2
9	CUSTOMER 9	2
10	CUSTOMER 10	2
11	CUSTOMER 11	2
12	CUSTOMER 12	2
13	CUSTOMER 13	2
14	CUSTOMER 14	2
15	CUSTOMER 15	2
16	CUSTOMER 16	2
17	CUSTOMER 17	2
18	CUSTOMER 18	2
19	CUSTOMER 19	2
30	CUSTOMER 30	2

NOTE: Product Line 1 refers to MOD3 associated projects whereas Product Line 2 refers to computer associated projects.

COST SAVINGS FEATURES

There are several cost savings features built into the system.

- Stacked input
- N-tasks
- Batch input
- Print options

Stacked input

A user, familiar with the system, and aware of the sequence of prompts, can input several responses at one time separated by diamonds "◇".

N-tasks

There is a significant cost saving achieved by running reports as an N-task; i.e., processing is delayed until the computer has a reduced work load. The delay is a function of the usage throughout the Sybron system and can vary between 1 second and a minute or two (usually the former). This delay doesn't preclude usage of the system whilst the N-task is being processed.

The cost savings of an N-task are threefold:

- There are no character transmission charges.
- The internal charges are less than half that of a terminal task.
- An N-task involves no terminal connect time.

Batch input

Processing some of the forecast data is more cost effectively handled by a special input routine, that minimizes input and validates the data at one pass. The example below illustrates all the features of Batch input.

The system is looking for three fields: *REF*, *VALUE*, and *MONTH*. These must be entered for each record. These three fields should be separated by commas. Separate records are separated by semi-colons. Brackets around *VALUE* and *MONTH* signify that these fields can be elided and the system will automatically assign the last previous value. (N.B. At least the first record needs a value for all 3 fields.) The 1: indicates that the system is waiting for input for the first record. The system will accept records and prompt for next record after every carriage return. The keyword "END" will halt this prompting sequence.

SAMPLE DATA

<u>RECORD</u>	<u>FIELD 1</u> <u>REF</u>	<u>FIELD 2</u> <u>VALUE</u>	<u>FIELD 3</u> <u>MONTH</u>
1	23	5	8206
2	24	4	8207
3	25	4	8207

4	26	10	8207
5	27	10	8208
6	28	14	8207
7	29	23	8206

SAMPLE TERMINAL SESSION (to input sample data).

<i>REF</i> ,[<i>VALUE</i>],[<i>MONTH(YYMM)</i>]; <i>REF</i>	ETC.
1:23,5,8206;24,4,8207	Carriage Return
3:25,,;26,10,;27,,8208;28,14,8207	Carriage Return
7:29,23,8206	Carriage Return
8: <i>END</i>	

The 7 records could have been entered on one line:

1:23,5,8206;24,4,8207;25,,;26,10,;27,,8208;28,14,8207;29,23,8206;*END*

Print options

We have the option of printing reports at the terminal or the computer center.

It is much more cost effective to have reports printed at the computer center, but sometimes a report is needed immediately and the terminal option is essential.

CONCLUSION

In conclusion, it can easily be stated that to be in business you need a marketing and manufacturing strategy. To be in marketing you need research data on sales potential and market trends. To be in manufacturing you need a master production schedule comprised of production expectations and an anticipated build schedule. And to remain competitive, you need to control, forecast and manage this data in a professionally accurate and cost efficient manner.

You need software to predict hardware.

PETROLEUM RESERVOIR SIMULATION IN APL

Rodman Jenkins
Chairman
Tejas Simulation International, Inc.
Dallas, Texas

Petroleum reservoir simulation is the calculation of the underground flows of oil, gas, and water, and sometimes heat, in oil-bearing rock or sands as oil and gas are produced from them. It is one of the largest industrial users of floating point arithmetic oriented ("number crunching") computing, and is the application area accounting for the largest non-government group of installations of vector processing (Cray, Cyber, etc.) super-computers.

It is currently a rapidly growing area because of the continuing world energy problem, and the gathering momentum of enhanced oil recovery projects which this has produced. On the average, less than one-fourth of the oil in a reservoir can be produced by primary methods—flowing or pumping—and fluids, heat, chemicals or some combination of these must be injected to produce part of the remainder. These enhanced recovery methods are more expensive and complicated, and therefore justify extensive computer analysis to make them more cost-effective.

Traditional FORTRAN territory

Until recently, there has been practically no APL use in this area. Most programs are in FORTRAN, with occasional examples in PL/1. The cost of using commercial time sharing services has been prohibitive, and companies with in-house APL systems also have existing libraries of FORTRAN routines to build on. These factors are also supported by two other considerations—the interpretation overhead of APL, and the recent development of vector computers, which do not yet offer APL, and may interpret inefficiently.

The opportunity for APL

Three types of opportunities for APL to penetrate this apparently unfriendly territory stand out. They are as follows:

1. Semi-interactive use. Simulators are currently run almost entirely in batch mode. As their complexity increases, so does the advantage of monitoring and interaction by a human being. APL is unsurpassed for such work.

2. Interfacing to user-written routines. We frequently find users (even management) performing extensive manual calculations on input or output for computer runs such as simulators produce. Such calculations are usually not highly structured, and unsuitable for FORTRAN, etc., but ideal for APL.
3. Development of superior algorithms. The efficiency of the procedures used here, particularly for enhanced recovery, is clearly limited by the complexity of the programs. Many of these programs represent investments of 10 or more man-years, and cannot be easily modified or rewritten, even though very large improvements in CPU usage are achievable. An APL system offers almost an order-of-magnitude reduction in the cost of testing the many potential new models and solution methods. In addition, some potential techniques use arrays of dynamically varying dimensions, which would give APL an inherent advantage.

The nature of enhanced recovery

Most enhanced recovery methods are displacement processes. That is, steam, water, carbon dioxide, air, or some other fluid is injected into a well to displace oil into nearby producing wells. Therefore, there will usually be at least three zones in the area being treated, as follows:

1. Displaced zone, containing little or no residual oil.
2. Frontal zone, where the injected fluid meets movable oil.
3. Producing or drainage zone, where displaced oil flows into the producing wells.

Flows in these zones will obviously be dominated by different physical effects. However, most currently available simulators use the same variables and equations in all the grid blocks, and hence perform large amounts of non-significant calculations.

Some examples of enhanced recovery processes

There are several oil recovery processes of current or probable near-term future significance. These may be classified as follows:

1. Steamflooding and steam stimulation
2. Carbon dioxide flooding
3. In-situ combustion
4. Polymer and surfactant flooding

Steamflooding is currently most widely used, and usually is performed by generating steam on the surface at high pressure in oilfield steam generators, and injecting it into selected wells. Computer simulation of such projects has been common for several years. The effectiveness of steamflooding is largely due to reduction in viscosity of the oil by the higher temperature, as well as the pressure gradient maintained by the steam and hot water flow. Change in wettability of the rock may also be a factor.

Carbon dioxide flooding is similar in that large quantities of the gas are injected under high pressure. This may require transport of the gas by pipeline several hundred or even thousand miles. Wells are even being drilled just to produce underground sources of carbon dioxide for injection into other fields.

It acts both by reducing the viscosity of the oil and by swelling it as it dissolves. The carbon dioxide may be either partially or completely miscible with the oil, depending on pressure, temperature, composition of the oil, and presence of other gases in the carbon dioxide mixture. The carbon dioxide is likely to be mixed with natural gas or other volatile hydrocarbons, so that calculation of the varying concentrations of these components as they migrate through the reservoir will be required of the simulator.

In-situ combustion is the process on which we have been testing our APL simulator, and is more complicated than the others because a combustion reaction between oil and injected air or oxygen generates heat to reduce oil viscosity and volatilize some of the oil. Figure 1 shows an artist's conception of in-situ combustion, showing the different zones which occur in the well pattern being treated.

In-situ combustion has been in commercial use since the 1950's, and large scale projects are underway in Canada, Europe, and South America, as well as in the U.S. We expect it to grow because most of its energy consumption is supplied by residual oil that would otherwise not be produced.

Polymer and surfactant flooding are processes in which the oil is displaced by water containing chemicals that reduce its surface tension or increase its viscosity. Both of these effects increase the amount of oil that will be displaced. Since all oil reservoirs also contain water, the concentration and effectiveness of these chemicals will be reduced as they are diluted by the formation water. The use of computer simulation to optimize the concentration of the additives used as well as other operating variables will also be vitally important in these processes.

What do I mean by a "simulator"?

Any physical or mathematical model that in some way represents some significant aspect of a petroleum reservoir's behavior could be described as a simulator. This would include laboratory tube studies, analog computers, or simple programs for hand held calculators.

However, in reservoir engineering, the word "simulator" has taken on a special meaning, to designate large programs that integrate the calculation of many or most of the physical and chemical laws governing the flow and composition of reservoir fluids, together with supporting routines for management, input and presentation of data from the large files involved in these calculations.

Physical and chemical principles

The basic physical rules involved in simulation of primary oil production are well known to petroleum technologists. They are:

1. D'Arcy's Law—The flow rate of a homogeneous fluid through a porous medium is proportional to the pressure or hydraulic gradient, and inversely proportional to the viscosity of the fluid.
2. Relative Permeability—In rocks or sands containing more than one fluid, the flow rate of each phase will be a fraction of what it would be if that were the only fluid present, and this fraction is a function of the proportion of each fluid present.

3. Volumetric and Phase Behavior—The relationship of vapor and liquid densities and compositions to pressure, temperature and composition of the phases with which it is in equilibrium. The effects of these variables on viscosity and capillary pressure may also be used.

Enhanced recovery simulation involves other physical and chemical processes and principles. Foremost among these are the following:

1. Heat conduction—Rate of heat flow is proportional to temperature gradient and thermal conductivity, which will be a function of local fluid content of the rock.
2. Vaporization, Condensation, and Distillation Effects—These are based on the kinds of equations for volumetric and phase behavior, but are more complex because of the greater variety of materials involved.
3. Combustion Chemistry—The quantity of combustion products formed, and quantity of heat released is related to the amount of oxygen and fuel consumed, and may also include the dependence of reaction rate on temperature, pressure and composition.

The additional factors involved in enhanced recovery processes make the simulator programming much more difficult, and also greatly increase the amount and cost of computer time required. The larger number of physical effects that may be considered increase the potential for APL programming, since it provides greater flexibility, and permits the simulator to be tailored to the particular effects that need to be considered in specific cases.

How reservoir simulators work

While alternative approaches are occasionally used, most simulators work by linearizing the flow equations for each identified component and heat with respect to the unknowns involved. A component may be a chemical compound, such as carbon dioxide or water, the entire oil phase, or some fraction of the oil defined by its volatility. The reservoir, or well pattern being analyzed is broken up into a grid (one, two, or three dimensional) which may be either rectangular or curvilinear, and is frequently non-uniform. The result is a large number of simultaneous, linear equations, typically two to seven per grid block. The number of equations to be solved will range from fewer than 100 to around 10,000.

Each solution of these equations represents a small time interval in the producing life of the well pattern or reservoir, and a full run may take from a few hundred to several thousand time steps. If all the equations are solved simultaneously, the simulator is called fully implicit. However, in some cases, a pre-elimination step is performed to eliminate some of the unknowns on an individual grid-block basis. The simultaneous solution is then much faster because of the smaller number of unknowns. On the other hand, it may require reduced time step size if there are significant interactions between the pre-eliminated unknowns. These are called explicit, since they are explicitly calculated from the other unknowns at the end of the time step.

Solution of these simultaneous equations is the most costly, CPU intensive part of the process. Both matrix inversion and iterative methods are used, and very complex coding is normally involved.

The TSI simulator

In order to investigate this application area, we have written a simulator in APL using essentially the same calculation procedure published for a previous FORTRAN in-situ combustion and steamflood simulator [Ref. 1]. Our simulator has been operational for several months, and we have made several runs that appear to give similar results to those from the FORTRAN simulator. The quantity and complexity of the input data required is so great that we have not yet attempted an exact match.

We have chosen to duplicate an existing algorithm rather than implement one designed to take immediate advantage of APL in order to get a good feel for the "architecture" of the problem, and to get at least an approximate validation of the result. As of this writing, therefore, our APL code uses more CPU time than the comparable FORTRAN version, by approximately a factor of ten.

We expect to be able to overcome this disadvantage quickly by one or more of several means, such as the following:

1. Variable switching
2. Reducing level of implicitness in some grid blocks
3. Sharing variables with FORTRAN routines

Grid blocks in the burned region contain a variable (oxygen concentration) that is inactive in the unburned region, and blocks in the unburned area contain a variable (gas saturation) that does not change in the burned zone. Exchanging these will reduce the total number of variables per block from 7 to 6, with roughly a factor of two reduction in CPU time.

Implicit variables are those whose solution is obtained simultaneously; explicit variables are derived from these on an individual grid block basis. CPU time usage is primarily dependent on the number of implicit variables, since these are part of the solution matrix. The number of variables that need to be implicit depends on what is going on in each individual block. FORTRAN code using this principle is currently becoming available [Ref. 2], and it should be much easier to implement in APL.

Some parts of the code, particularly matrix inversion, use a large part of the CPU time, are standard in format, and could be done faster by a FORTRAN routine. We are currently on an APL system having shared variables (Dome/Sharp) and are therefore considering this approach to reducing CPU usage.

Implementation effort to date has been approximately two man-years. While I do not know the effort required for comparable FORTRAN programs, I believe it to be several times higher. In this kind of problem, the cost-effectiveness of the program is directly related to the complexity of the algorithm, so I expect this productivity to translate itself into a cost-effective program product.

Use of vector processing

The number of vector processing computers (Cray, Cyber, FPS) currently being installed is clear evidence of the acceptance and cost-effectiveness of this approach for arithmetic-intensive problems. There is a clear need for APL users to be able to take advantage of such hardware. I will leave it to listeners and readers to develop appropriate methods for achieving this goal.

Sample problem and output

We have been testing our simulator on a field case published with the algorithm we have been following [Ref. 1]. This case represents an actual test of oil recovery by in-situ combustion performed by the Sinclair Oil and Gas Co. in Nowata County, Oklahoma, from 1960 to 1965, which has been described in a paper in the *Journal of Petroleum Technology* [Ref. 3].

The grid used in this test problem is shown in Figure 2, and consists of four vertical layers of 20 grid blocks each. This represents a 45° wedge segment of a symmetrical "developed five-spot" well pattern, which essentially assumes an infinite field of alternating injection and producing wells in both directions, all operated simultaneously, so that there is no flow between the individual patterns. The layers have different permeabilities, which also have different values in the vertical and horizontal directions.

A typical simulator run will take 1500-2000 time steps, and values of seven independent variables (pressure, temperature, gas saturation, water saturation, and concentration of oxygen, nitrogen and carbon dioxide in the oil) will be determined in each step. Printout is made each tenth time step, which results in 150-200 pages of output. A sample page is shown in Figure 3.

Typical curves of oil production rate and air injection rate are shown in Figure 4. The "bumps" in the curves are probably due to the finite size of the grid blocks used, and might not be noticeable if the grid could be expanded to several hundred blocks. Contour plots are frequently used in presenting such results. A typical water saturation contour plot is shown in Figure 5. Again, the "squared-off" shape of the inner contours is due to the limited number of blocks which currently can be used, which is typical of simulators of this kind.

As the number of grid blocks increases, the percent of CPU time used for interpretation will decrease, so that APL will become relatively more efficient as larger problems become practical.

APL code

The running simulator consists of about 30 pages of APL code, and we have approximately 10 additional pages of auxiliary functions for set-up, monitoring and diagnosis. Symbol table size is 460, although we hope to reduce this to make maintenance and training easier. It has been developed with a 256K workspace size, but currently uses about a million bytes of working storage on files. CPU usage is almost entirely processing, and is relatively insensitive to our file usage. We use packages with the files, and find them to be quite helpful.

Most FORTRAN simulators are run in partitions much larger than this workspace, so that we don't expect file handling to be much of a future problem. However, the ability to use a 256K workspace has been excellent for development purposes, since we have been able to get a large application running on a system without significantly impacting other users.

A sample page of code is shown in Figure 6. This function calculates parameters called accumulation coefficients, which are an array of rank 5. It is mostly straightforward array arithmetic, with references to several other functions, which primarily calculate physical properties. It uses rotate and transpose, which also occur frequently in other parts of the code.

The most difficult parts of the application are indexing problems relating to selection of upstream and downstream grid blocks for flow of the three different phases, and location of terms within the solution matrix.

Program operations

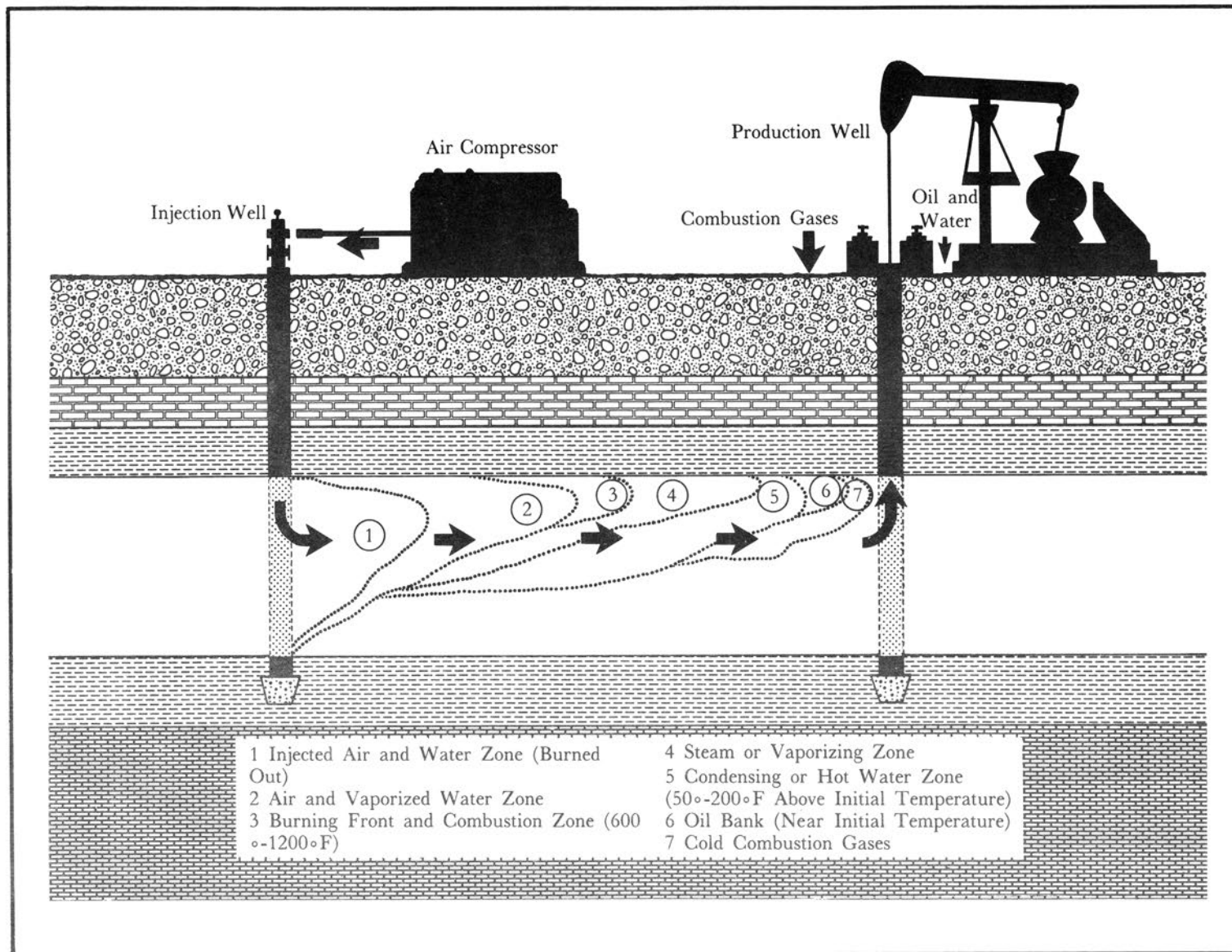
Runs are normally started as T-tasks, and monitored for 10 to 20 time steps to confirm proper operation. This is always desirable with simulators of this kind, because with the large variety and volume of input data used, it is easy to have errors present—even with very careful editing—and because of the substantial CPU usage per run. We currently avoid continuous running on day shift, but use N-tasks for evening runs because of ease of interruption and restart, and convert to B-tasks for continuation overnight. Two overnight sessions are typically required for completion of a run (the system is normally down from midnight to 3:00 a.m., and lightly loaded from then until morning). Changes currently being made will make a major reduction in running time.

Conclusions

APL has been a cost-effective tool for development of prototype code for a very complex, computation-intensive application. It provides a variety of opportunities for development of advanced algorithms that will compete successfully with other languages in production run costs as well as user convenience and integration with other facets of the user's work.

References

1. Youngren, G. "Development and Application of an In-situ Combustion Reservoir Simulator", *Society of Petroleum Engineers Journal*. Feb. 1980, pp. 39-51.
2. Thomas, G.W. "The Mathematical Basis of the Adaptive Implicit Method", Society of Petroleum Engineers, Paper 10495, *Sixth SOCIETY OF PETROLEUM ENGINEERS Symposium on Reservoir Simulation*. New Orleans, Feb. 1982.
3. Barnes, A.L. "Results From a Thermal Recovery Test in a Watered-out Reservoir", *Journal of Petroleum Technology*. Nov. 1965, pp. 1343-1353.



In-situ Combustion

Figure 1

Figure 2
Grid For Three-Dimensional Simulation

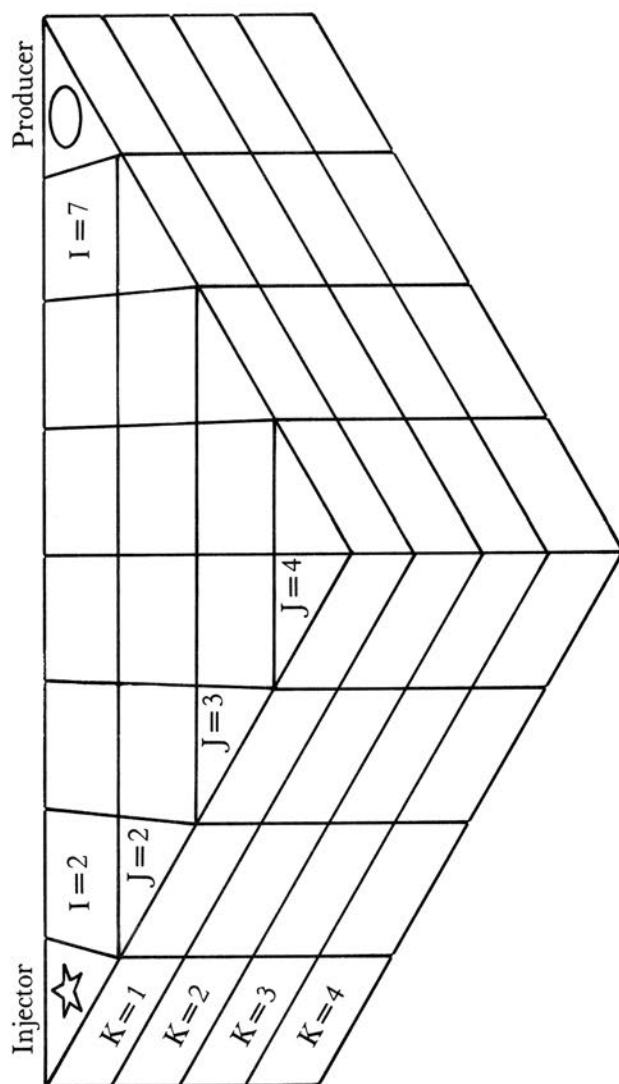


Figure 3

TIME= 1000										ELAPSED TIME= 695.257									
PRODUCTION RATES					WOR		GOR		CUM PRODUCTION			INJ RATES			CUM INJECTION				
OIL	GAS	WATER	STB/	MCFD/	OIL	GAS	WATER	WATER	GAS	OIL	GAS	WATER	WATER	GAS	WATER	GAS			
B/D	MCFD	B/D	STB	STB	MSTB	MSCF	MSTB	STB/D	MCFD	MSTB	MSCF	MSTB	STB/D	MCFD	MSTB	MSCF			
7.5	78.5	7.5	10.4	1.0	2.0	27119.	7.8	0.0	91.1	0.	30222.								
OILBAL= 1.01731 H2OBAL= 0.99732 N2BAL= 0.98421 HTBAL= 0.98941 HTLOSS= 3903840.6																			
PRESSURE, PSIA										TEMPERATURE, DEG F									
485.29	482.67	480.39	478.23	473.75	459.02	441.92	404.09	274.38	438.19	504.27	425.47	342.89	183.23	134.83	126.56				
	481.53	480.31	477.34	471.94	464.05	453.36			544.49	389.25	250.91	153.80	127.20	122.73					
		478.37	476.06	472.72	468.32					215.33	141.20	123.90	121.31						
			474.95	473.64							123.16	120.82							
485.49	481.78	479.22	476.80	472.81	458.05	441.39	401.87	270.87	465.94	621.66	429.29	380.07	238.04	161.95	141.08				
	480.52	479.40	476.13	471.54	464.53	453.79			640.37	397.00	281.50	168.60	132.86	125.76					
		477.25	475.59	472.41	467.86					243.75	150.58	125.95	122.16						
			474.53	473.17							124.65	121.23							
476.17	474.10	471.76	468.79	464.55	458.55	447.71	420.90	246.46	431.23	529.63	572.44	641.75	359.76	272.52	195.15				
	473.05	471.65	470.27	466.01	461.60	455.54			533.94	526.40	339.99	199.77	143.50	132.79					
		471.36	469.84	467.70	464.78					291.59	164.92	128.56	123.24						
			469.36	468.48							126.21	121.58							
475.48	473.45	471.30	468.64	465.03	460.40	447.69	426.42	237.79	394.05	474.85	542.55	599.96	365.94	248.10	176.86				
	472.53	471.39	470.51	466.65	462.90	456.26			475.05	444.45	340.43	187.58	140.69	130.76					
		471.26	470.33	468.54	465.93					273.98	154.95	126.84	122.69						
			470.04	469.33							124.75	121.24							
GAS SATURATION										WATER SATURATION									
1.0000	1.0000	1.0000	0.9138	0.2421	0.1932	0.1591	0.1508				0.0436	0.6268	0.5842	0.4746	0.4508				
	1.0000	0.4067	0.2374	0.1981	0.1686	0.1651				0.5024	0.6232	0.4821	0.4371	0.4316					
		0.2482	0.1952	0.1750	0.1762					0.5658	0.4483	0.4401	0.4523						
			0.1741	0.1956							0.4434	0.4715							
1.0000	1.0000	0.9433	0.6579	0.2353	0.1927	0.1702	0.1634				0.2623	0.6325	0.6642	0.5058	0.4558				
	0.9454	0.3240	0.2151	0.1515	0.0950	0.1019				0.5446	0.6538	0.5141	0.4719	0.4823					
		0.2224	0.1177	0.0808	0.1044					0.6301	0.4834	0.5228	0.5588						
			0.0793	0.1039							0.5313	0.5747							
1.0000	1.0000	1.0000	0.9997	0.9781	0.4409	0.2602	0.2452						0.4888	0.6058	0.5599				
	1.0000	0.9699	0.3130	0.2324	0.1792	0.1587					0.5554	0.5740	0.4881	0.4614					
		0.2689	0.2023	0.1567	0.1499					0.5969	0.5242	0.4571	0.4502						
			0.1538	0.1697							0.4548	0.4707							
1.0000	1.0000	1.0000	1.0000	0.9541	0.3612	0.2339	0.2047						0.5205	0.6287	0.5525				
	1.0000	0.9585	0.2402	0.1628	0.0973	0.0729					0.6275	0.6191	0.5189	0.4877					
		0.2008	0.1158	0.0525	0.0425					0.6718	0.5258	0.4826	0.4867						
			0.0441	0.0351							0.4883	0.5580							
MOL FRAC STEAM										MOL FRAC NITROGEN									
0.0004	0.0008	0.0009	0.6873	0.2597	0.0177	0.0058	0.0050	0.7899	0.7898	0.7898	0.2696	0.6379	0.8461	0.8561	0.8566				
	0.0009	0.4571	0.0636	0.0087	0.0045	0.0041			0.7898	0.4682	0.8078	0.8544	0.8576	0.8578					
		0.0329	0.0063	0.0040	0.0038					0.8348	0.8566	0.8579	0.8580						
			0.0039	0.0037							0.8579	0.8585							
0.0003	0.0010	0.1280	0.7185	0.4143	0.0527	0.0113	0.0074	0.7899	0.7898	0.7516	0.2430	0.5042	0.8135	0.8477	0.8509				
	0.1222	0.4996	0.1060	0.0123	0.0052	0.0044			0.7567	0.4308	0.7700	0.8491	0.8554	0.8563					
		0.0561	0.0079	0.0043	0.0039					0.8127	0.8519	0.8568	0.8571						
			0.0041	0.0037							0.8563	0.8584							
0.0001	0.0003	0.0174	0.1604	0.2847	0.3328	0.0976	0.0249	0.7900	0.7899	0.7848	0.6827	0.6234	0.5823	0.7865	0.8489				
	0.0178	0.1930	0.2509	0.0246	0.0069	0.0053			0.7852	0.6957	0.6460	0.8405	0.8552	0.8562					
		0.1253	0.0114	0.0046	0.0040					0.7539	0.8513	0.8564	0.8567						
			0.0043	0.0038							0.8565	0.8568							
0.0002	0.0007	0.0334	0.1781	0.3300	0.3599	0.0646	0.0165	0.7900	0.7898	0.7664	0.6560	0.5777	0.5520	0.8061	0.8470				
	0.0022	0.3812	0.2524	0.0191	0.0064	0.0050			0.7894	0.5380	0.6496	0.8462	0.8544	0.8553					
		0.0950	0.0090	0.0044	0.0039					0.7818	0.8522	0.8558	0.8563						
			0.0041	0.0037							0.8564	0.8574							
MOL FRAC CO2										MOL FRAC O2									
0.0004	0.0009	0.0010	0.0431	0.1024	0.1363	0.1381	0.1383	0.2094	0.2085	0.2083									
	0.0010	0.0747	0.1287	0.1369	0.1379	0.1382				0.2083									
		0.1323	0.1371	0.1381	0.1382														
			0.1382	0.1379															
0.0004	0.0010	0.1204	0.0386	0.0814	0.1338	0.1410	0.1417	0.2094	0.2083										
	0.1211	0.0695	0.1241	0.1386	0.1394	0.1393													
		0.1312	0.1401	0.1389	0.1390														
			0.1396	0.1379															
0.0001	0.0003	0.0165	0.0274	0.0919	0.0850	0.1159	0.1263	0.2098	0.2095	0.1813	0.1295								
	0.0177	0.1113	0.1031	0.1349	0.1379	0.1385			0.1794										
		0.1208	0.1373	0.1390	0.1392														
			0.1392	0.1394															
0.0002	0.0007	0.0044	0.0108	0.0923	0.0881	0.1294	0.1365	0.2103	0.2088	0.1959	0.1551								
	0.0022	0.0808	0.0981	0.1347	0.1392	0.1397			0.2062										
		0.1232	0.1388	0.1398	0.1397														
			0.1394	0.1388															

Figure 4

Air Injection and Oil Production vs Time

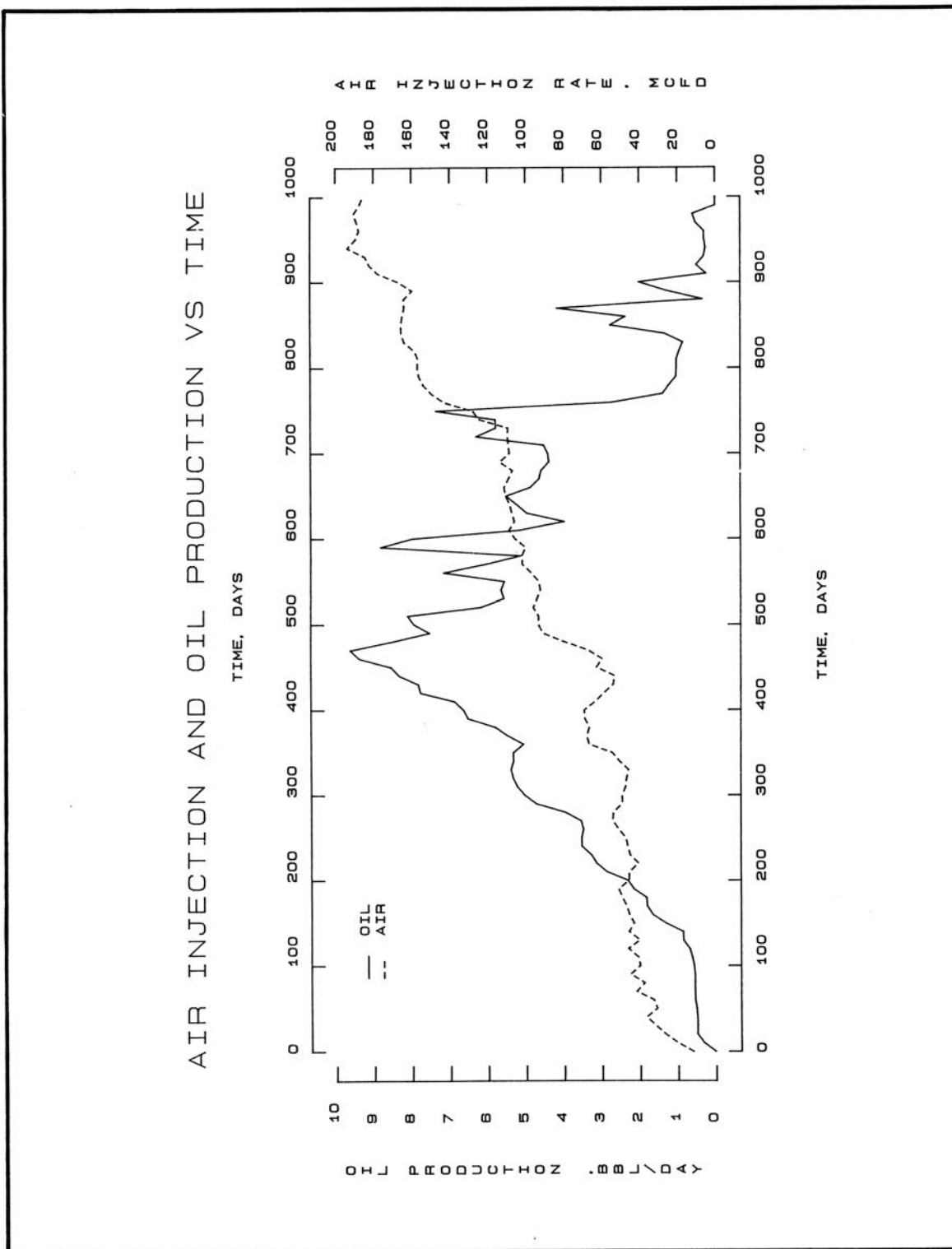
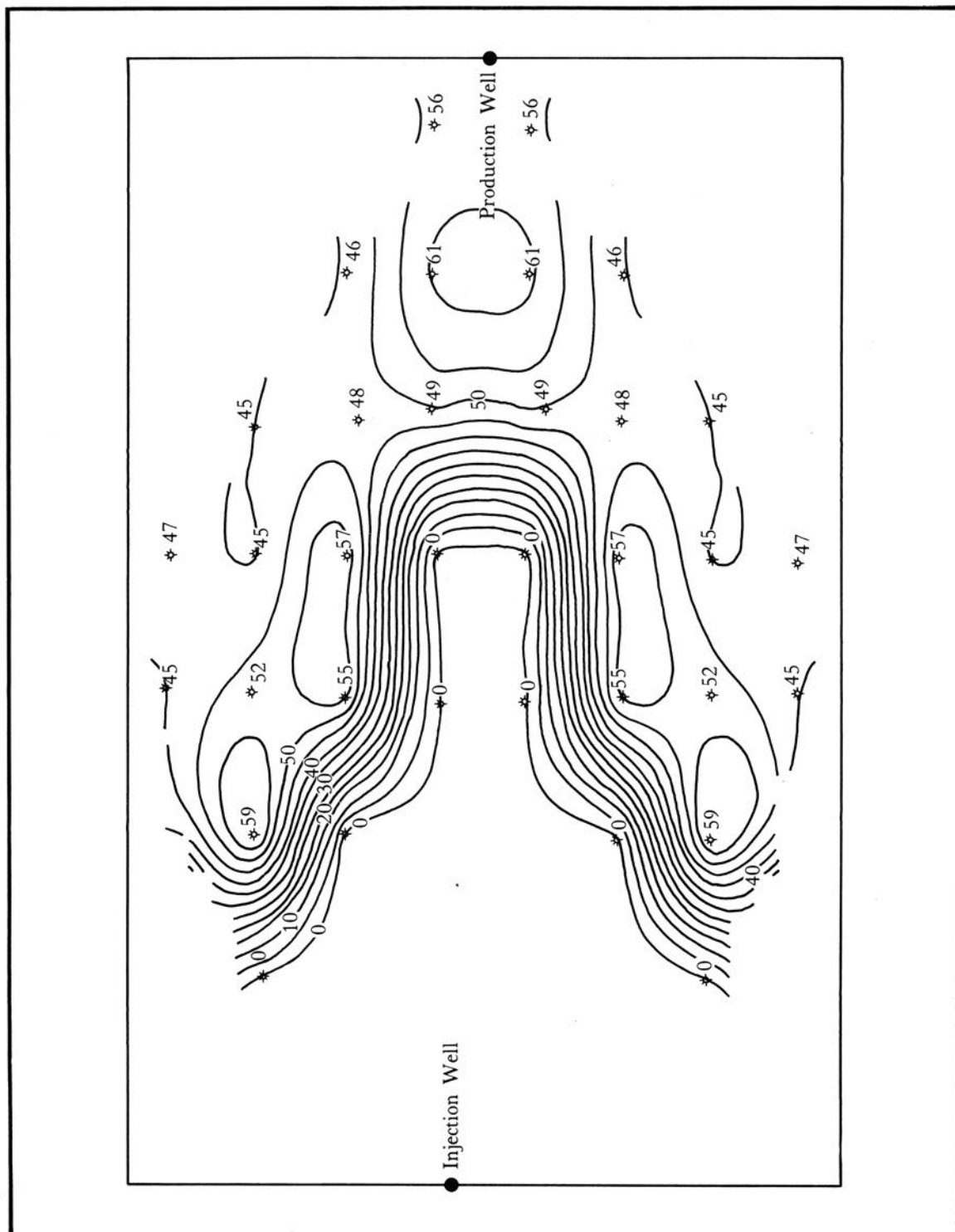


Figure 5

A Real Water Saturation Contour in Layer 3
(Run 1 at 749 days)



```

▽ CCOF2[ ]▽
▽ CCOF2;RPP;GPP;N;Q;S;B;W;SRSO;ARG;SWCT;SWR;RHOH;ASGH;BPC;KVRG;XLA;SGH;RGPP;I;V
[1] C←(7 7 ,DM2)ρ0
[2] A←PHI×B+VOL÷DELT
[3] C[1;1;;;]←(ARG+A×RGOLD+RHG)×PP+YS
[4] C[1;2;;;]←A×RHW×H
[5] C[1;6;;;]←A×((GPP+SG×PP)×RHGT)+(V+SG×RHG)×PSTP+PST÷P)-RHW×SWCT+CTW×SW
[6] C[1;7;;;]←(B×PHICR×SWR+RGPP)+A×(CW×SWR+SW×RHW)+(GPP×RHGP)-(RGPP+RHG×GPP)÷P
[7] C[N+1+15;1;;;]←(XLA+XL×(QρA))×(KVRG+(KV+KVL)×QρRHG)-(Q+ 4 1 ,DM2)ρRHO
[8] C[N;2;;;]←(XLA+XL×QρA)×QρRHO×H
[9] W← 3 1 ,DM2
[10] C[2; 3 4 5 ;;;]←-(1,DIM)ρSRSO+A×RHO×SO+1-SW+SG
[11] C[I;I+ 3 4 5 ;;;]←(2 3 4 1 ϕ(DM2,3)ρ- 0 1 2)Φ[2](((KV[1+14;];;])×WρA×V)+RGXI×(WρXLA)×WρSG)+WρSRSO),[2](3 2 ,DM2)ρ0
[12] C[N;6;;;]←(XLSG+(QρXLA)×QρSG)×(KIT×QρRHG)+KV×QρRHGT)-XLA×QρSO×CTO×RHO
[13] C[N;7;;;]←(XLSG×(KIP×QρRHG)+KV×QρRHGP)+(XL×(QρSG×BPC+B×PHICR)×KVRG+KV×QρRHG)+XLA×QρSO×CO×RHO
[14] C[6;1;;;]←(ARG×HGT+PP HG T)-A×RHOH+RHO×HOIL+XL[1;1;;;] HO T
[15] C[6;2;;;]←H×(C[1;2;;;]×HWAT+HW T)-RHOH×A
[16] C[6; 3 4 5 ;;;]←((DIMρARSG+ARG×SG)×HGK)+(DIMρASGH+A×SGH+SG×HGT)×RGXI
[17] C[6; 3 4 5 ;;;]←C[6; 3 4 5 ;;;]+DIMρSRSO×(CPG-CPO)×T-TI
[18] C[6;6;;;]←(C[1;2;;;]×SW×CPW)+A×(-SWCT×HWAT×RHW)+SO×RHO×CPO-HOIL×CTO
[19] C[6;6;;;]←C[6;6;;;]+(ARSG×CPG)+(ASGH×RHGT)+B×(1-PHI)×CPR
[20] C[6;7;;;]←(BPC×(SWR×HWAT)+(SO×RHOH)+SGH×RHG)+(A×(SWR×CW)+(SO×RHO×CO)+SGH×RHGP)+B×CR×HROCK×1-PHI
[21] C[7;2;;;]←~H
[22] C[7; 3 4 5 ;;;]←KV[2 3 4 ;1;;;]
[23] C[7;6;;;]←(+/(XL[1;1;;;]×KIT[1;1;;;]))+H×PSTP
[24] C[7;7;;;]←(+/(XL[1;1;;;]×KIP[1;1;;;]))-H×PP÷P
[25] CDRY
[26] CINJ
▽

```

Figure 6

EXCEED - EXECUTIVE MANAGEMENT DECISION SUPPORT SYSTEM

Hiroshi Isobe
Software Works
Hitachi, Ltd.
Yokohama, Japan
and
Kotaro Yamashita
System Development Laboratory
Hitachi, Ltd.
Kawasaki, Japan

1. INTRODUCTION

It is generally admitted that many data processing departments cannot meet the wide range of demands made by today's end users. The following two approaches are considered to be effective in solving this problem: 1) developing user-oriented systems which enable end users to directly manipulate information stored in the computer, and 2) developing application program development support systems which enable programmers in data processing departments to write application programs quickly.

It is necessary to implement both systems in order to meet user needs. However, conventional programming languages like COBOL and FORTRAN do not provide an effective means for creating such systems. I think APL offers an extremely effective way of implementing both types of systems, but many people do not agree with me. They offer the following objections:

- The number of programmers who are familiar with APL is very limited and—what is worse—almost all of the decision makers in data processing departments have never even heard of APL.
- Good APL systems are not widespread and many people mistake the effects of bad implementations for weaknesses in APL itself.
- It is easy to write and debug APL programs, but the performance of debugged programs is very poor.
- APL is weak in file processing.
- Data created by conventional languages like COBOL and FORTRAN, or data in a large scale data base system cannot be accessed from APL.
- It is a little difficult for end users to master APL.

- The exclusiveness of APL enthusiasts: some APL zealots believe APL is the only programming language worth mastering, and they usually do not explain their reasons for this position in detail.

Of course, APL has many advantages which COBOL and FORTRAN do not have. I shall not go into detail here because most of my readers already know them.

Because of the existing attitudes toward APL, I judged that it was not appropriate to consider APL itself as an end-user-oriented system or application program development support system. Rather, I decided to develop a new system which is based on APL in order to realize the two types of systems.

The system I developed is called EXCEED, which stands for Executive Management Decision Support System. The major design emphasis of EXCEED is to implement an end-user-oriented system, but I believe EXCEED is also an effective tool as an application program development support system. Our company contracted to use I.P. Sharp's software (MAGIC, MABRA, SUPERPLOT, SAGA) to implement EXCEED. In this paper, I shall describe the characteristics of EXCEED, its relationship to I.P. Sharp's software, and the considerations necessary to adapt it to the Japanese language.

2. BASIC CONCEPTS AND TERMINOLOGY

To make EXCEED truly easy for end users to use, I considered it essential for the system to reflect the manner in which its users normally do their office work. The basic data processing cycle of EXCEED is different from the traditional one which gets data from transaction files, updates a master file, and finally generates reports. EXCEED is a natural extension of office work; that is, users get *papers* from their *cabinets* and write onto those papers wherever they need to update or record anything.

Cabinets and record files

Cabinets and *record files* are provided for data storage. *Record files* are used to store large and uniform data such as the monthly sales values for the past two decades or the personnel information of big companies. On the other hand, *cabinets* are used to store small and diverse data like the reports for an executive meeting.

More precisely, there are two types of record files: *time series* and *general*. The time series record file is used to store data along a time axis (daily, monthly, and yearly). Its structure is almost equivalent to a MAGIC file. Each general record file represents one relation of a relational data base. The structure is similar to a MABRA system, but several facilities like field width (occurrence of field) and absolute record numbering are eliminated to ensure that it is a perfect subset of a relational data base. Figure 1 shows simple examples of record files.

Figure 1
Structure of Record Files

'SALES' file (Time Series type)

Field Names		Monthly data									
PRODUCT	REGION	1978/1	1978/2	1978/3	1978/4	1978/5	1978/6		1982/1	1982/2	1982/3
STEREO	HOKKAIDO										
STEREO	KANTO										
STEREO	KANSAI										
TV	HOKKAIDO										
TV	KANTO										

'PERSONNEL' file (General type)

Field Names					
NAME	EMPLOYEE NUM	DEPT NAME	BIRTH DATE	ENTRANCE DATE	SALARY

Desks

Desks are similar to APL active workspaces. All work is done after the objects to be processed are transferred (moved or copied) to a desk.

Tables

The concept of *tables* is abstracted from the reports and papers used in daily office work. A table is created from data in a record file or from data entered directly at a terminal and stored in a cabinet. Tables also have two types: *time series* and *general*. Figure 2 shows screen images of a times series type table and a general type table.

Figure 2
Screen Images of Tables

Time Series Type Table

MONTH	STEREO	
	HOKKAIDO(6)	KANTO(6)
1981/01	16.9	61.7
1981/02	17.3	59.1
1981/03	17.4	61.8
1981/04	17.1	60.3
1981/05	17.6	61.8
1981/06	17.2	62.4
1981/07	17.8	63.9
1981/08	17	65.1
1981/09	16.4	67.2
1981/10	17.7	62.9
1981/11	17.9	65.4
1981/12	18.6	70.2

General Type Table

NAME	EMPLOYEEENUM	DEPTNAME	BIRTHDATE	ENTRANCEDAT	SALARY
YAMADA TARO	4185	SALES	1949/10/03	1971/04/01	243200
NAKANO HITOSHI	3587	SALES	1943/06/04	1967/10/01	273080
INOUE JIRO	7583	SALES	1956/07/25	1978/04/01	197400
SASAKI HAJIME	4873	PLANNING	1938/01/24	1950/09/01	348700
MATSUNO TADASHI	2587	PLANNING	1960/08/06	1981/04/01	143700
KATO SHINJI	8521	PLANNING	1957/03/04	1979/09/01	187500

Sheet definition

A *sheet definition* contains the information which defines chart area names, the positions where table data will be displayed on a page, the type of output device to be used, and the sheet title text. Figure 3 shows an example of a sheet definition.

Figure 3

Example of a Sheet Definition

```
*** SHEET('SAMPLESHT') DEF. OPTION LISTING ***  
  
DEVICE: HP7221C           , KANJI  
POSITION: CHAREA1         5 1 50 40  
POSITION: CHAREA2         5 45 50 30  
TEXT:  SAMPLE TITLE BY GREEN AT 1 20 TO RIGHT  
  
*** END OF SHEET DEF. OPTION LISTING ***
```

Chart definition

A *chart definition* contains the information which defines how to display table data in tabular form or as business graphics. *EXCEED* defers correlating chart definitions with their respective chart areas in a sheet definition until the system actually generates a report. Figure 4 shows an example of a chart definition.

Figure 4

Example of Chart Definition

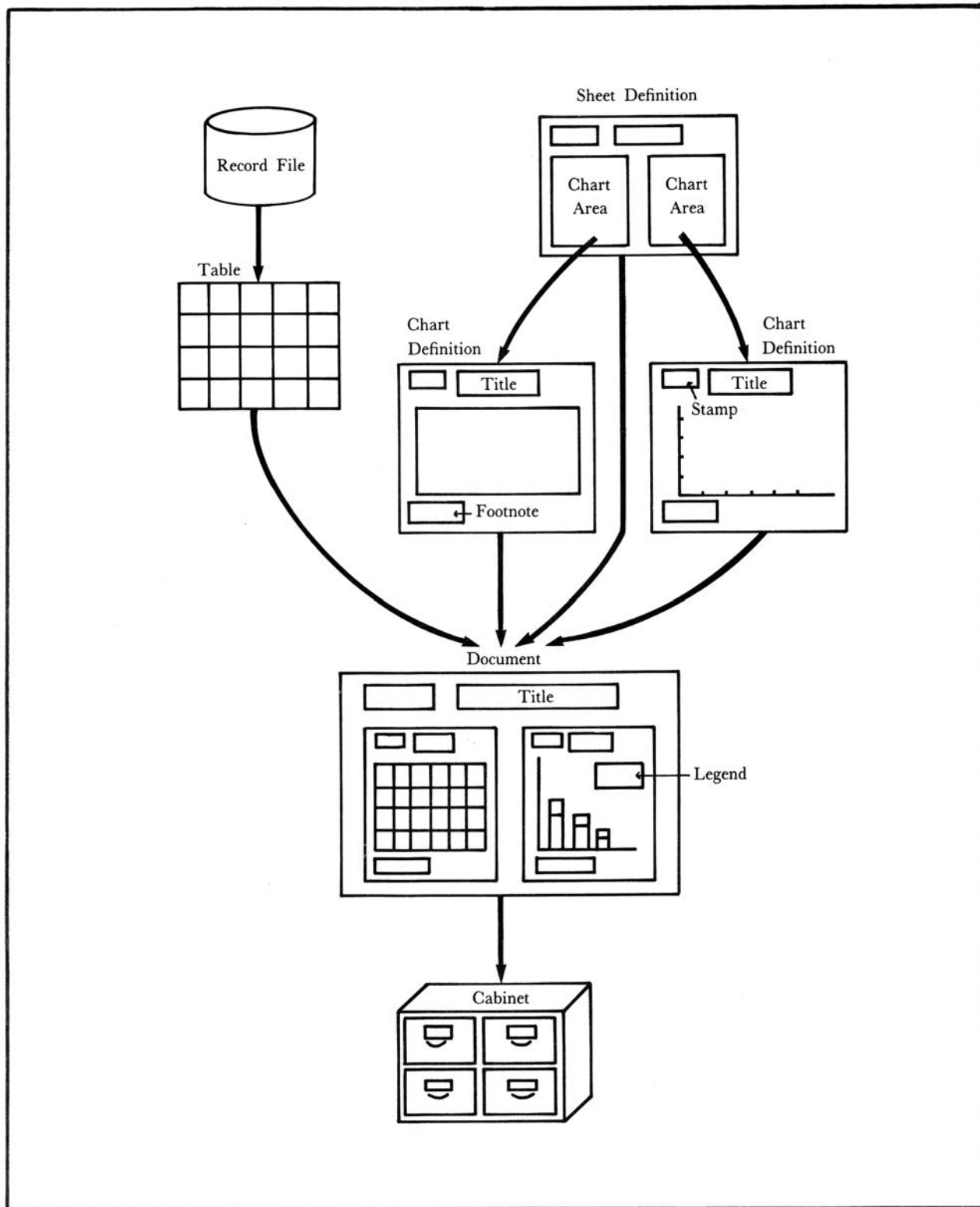
```
*** CHART('SAMPLEG') DEF. OPTION LISTING ***  
  
TYPE:BOX,BOX,BOX  
COLOUR:GREEN ,RED ,SKY  
SHADING:1,BOTTOM,45,10,GREEN ,SOLID  
SHADING:2,BOTTOM,45,10,RED ,SOLID  
SHADING:3,BOTTOM,45,10,SKY ,SOLID  
  
*** END OF CHART DEF. OPTION LISTING ***
```

Documents

Documents are created by transforming table data according to sheet definitions and chart definitions. If there is more data than will fit on a page, a document consists of multiple pages. If end users want to store transformed table data in cabinets, they must create documents. However, if they need not store them, they can display transformed data directly on terminal devices. Figure 5 shows the relationship between tables, sheet definitions, chart definitions, and documents.

Figure 5

Relationship Between a Document and Other Objects



Commands

The dialogue between end users and the computer is carried out with instructions called *commands*. There are two types of commands: *prompting* and *parameter*. With prompting commands, end users supply answers to system queries. With parameter commands, users specify all the necessary information after entering a command name. Examples of the two types of commands are shown in Figure 6.

Every command name consists of a two-character mnemonic code followed by an English verb. The mnemonic code represents the type of object on which the command acts. For example, *TA* stands for table and *CA* for cabinet. Plain English verbs are used as command names. Typical command names are *TACREATE* (create table) and *CADISPLAY* (display cabinet index).

Figure 6

Example of Two Types of Commands

Prompting Type

```
ENTER COMMAND: TACOPY
COPY TO DESK OR CABINET? (D/C) : D
ENTER CABINET NAME : CAB1
ENTER TABLE NAME(S) : URIAGE
'URIAGE' COPIED.
```

Parameter Type

```
ENTER COMMAND: TACOPY URIAGE FROM CAB1
'URIAGE' COPIED.
```

EXCEED users who have a full knowledge of EXCEED commands can create a new command by combining EXCEED commands. We call such users *EXCEED specialists*, and the commands they create are called *user commands*. Actually, bare APL expressions can also be included in a user command. Commands provided by the EXCEED system are called *basic commands*, to distinguish them from user commands.

Trace mode

To strengthen the analogy to normal office work, the concept of *trace mode* has been introduced. Since the table currently being used on a desk is displayed on the terminal screen in trace mode, end users can direct the computer while viewing a table. Figures 7 and 8 show the screen layout in trace mode and nontrace mode respectively. Figure 9 shows the transition of screen images with the *DAUNGROUP* command under trace mode.

Figure 7
Screen Layout of Trace Mode

EXCEED S1-00

SALES

82-7-5

MONTH	STEREO KANTO(6)
1981/01	61.7
1981/02	59.1
1981/03	61.8
1981/04	60.3
1981/05	61.8
1981/06	62.4
1981/07	63.9
1981/08	65.1
1981/09	67.2
1981/10	62.9
1981/11	65.4
1981/12	70.2

ENTER COMMAND: DAUNGROUP

.....

Current table name

Current table

Input area

Figure 8
Screen Layout of Nontrace Mode

EXCEED S1-00

Current position of scroll file

LINE 589 COL 1

SALES1

82-7-5

ENTER COMMAND: DAUNGROUP

ENTER TABLE NAME: SALES

ENTER GROUP NAME OR CLASS NUMBER FOR UNGROUPING: STEREO

ENTER RESULT TABLE NAME: SALES1

}

History of dialogue

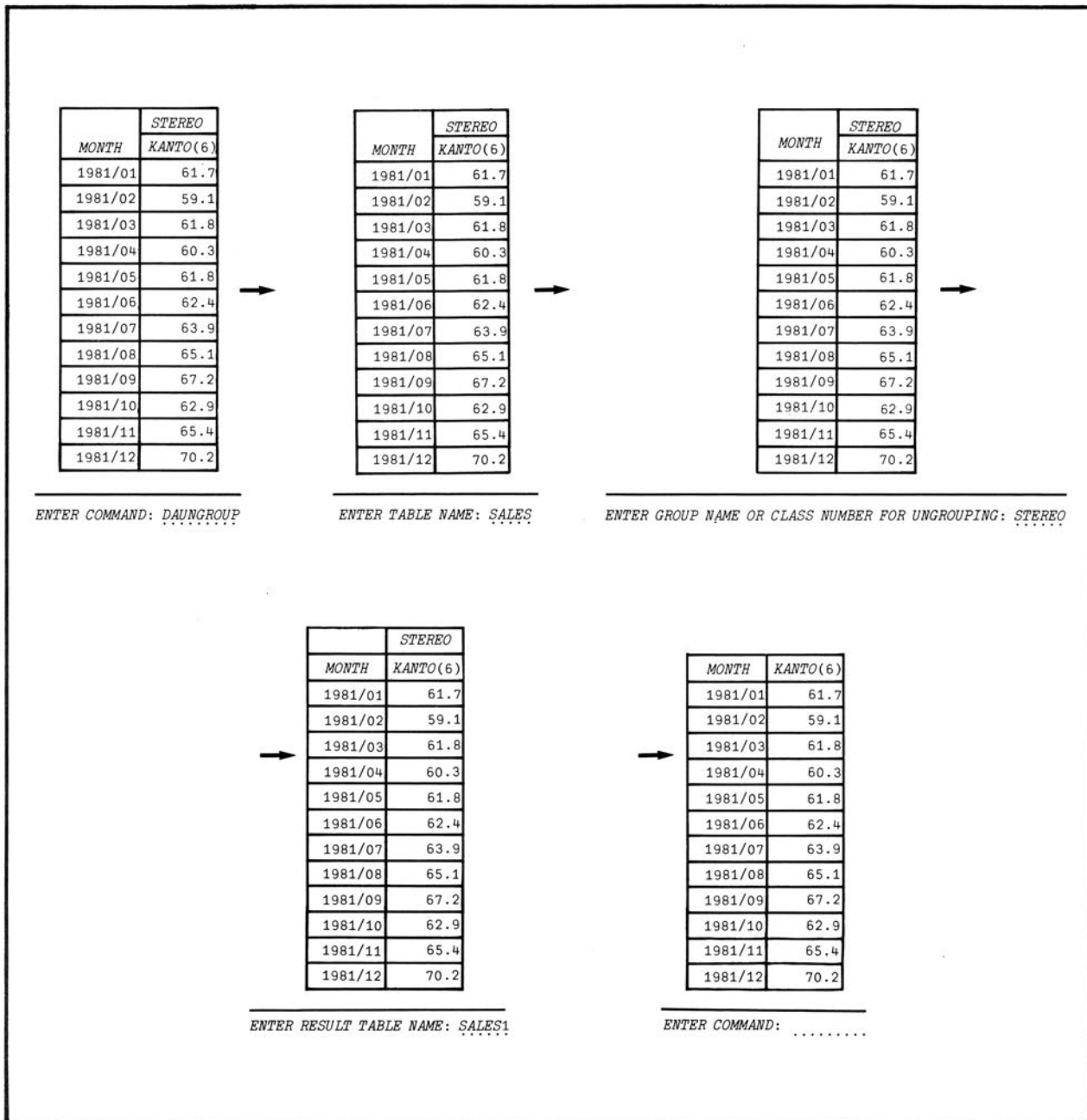
ENTER COMMAND:

.....

Input area

Figure 9

Transition of Screen Images in Trace Mode



Panel definition

When EXCEED specialists write user commands, they generally want to create a new man/machine interaction with its own specialized commands. In such cases, they design a screen layout and field attributes called *panel definitions*. EXCEED provides a facility to create panel definitions interactively. Figure 10 shows an example of a user-defined panel used for personnel information retrieval.

Figure 10

User-Defined Panel

人事情報検索 (NO3)							
検索条件							
24. 配置	(本人)	? 継続	? 部門内	? 他部門	? 社外	? どこでも	? その他
	(上)	? 継続	? 部門内	? 他部門	? 社外		
25. 職務	(本人)		(上)				
26. 勤務場所	(本人)	? 本社	? 営業所	? 研究所	? 工場	? 海外	
	(上)	? 本社	? 営業所	? 研究所	? 工場	? 海外	
27. 勤務地							
28. 現職開始日	年	月以降	年	月以前	年	月のみ	
29. 成 地							
30. 資格免許							
31. 語学能力	英語	独語	語	語	語	中国語	
	=	=	=	=			
32. 住 区分	? 自宅	? 社宅	? 独	? 会社施設	? その他		
33. 家 区分	? 有家	者	? 単 者				
34. その他条件							

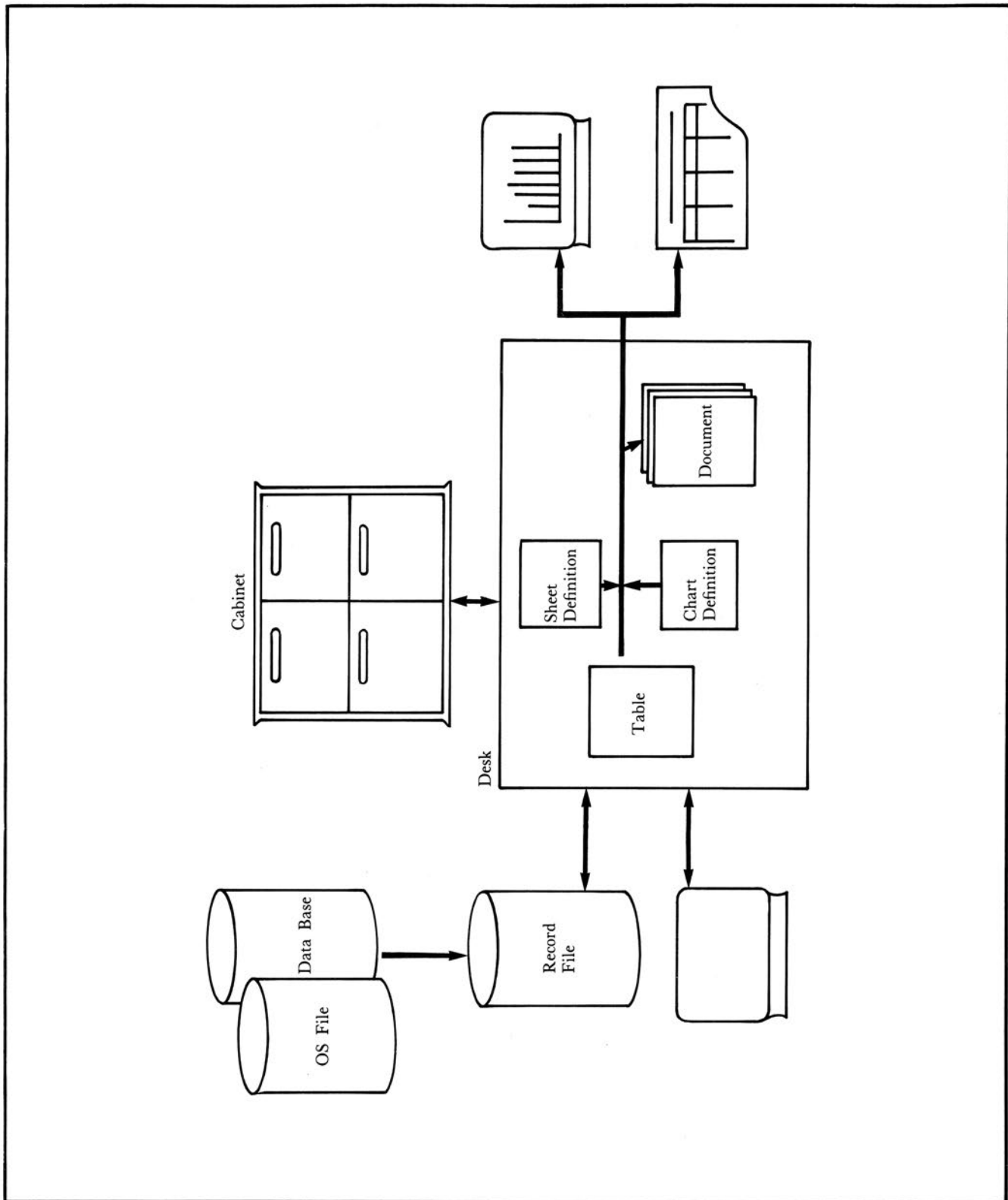
Command file

User commands, panel definitions, and APL objects employed in user commands are stored in *command files*.

Security considerations

It is important to provide a facility to secure information stored in an EXCEED system to prevent unauthorized people from obtaining access to it. Security protection should be set by the owner of the information without the help of data base specialists in data processing departments. A wide range of security levels for EXCEED storage (record files, cabinets, and command files) can be determined and set by the end users themselves. Figure 11 shows the basic data flow in EXCEED.

Figure 11
Basic Data Flow in EXCEED



3. CHARACTERISTICS OF EXCEED

The characteristics of EXCEED as an end-user-oriented system and application program development support system are described below:

(1) Characteristics as an end-user-oriented system

End users of EXCEED are not programmers in data processing departments. Rather, they are middle managers or specialists in offices. Therefore, I attached great importance to making EXCEED a truly "easy to use" system.

(a) **Easy data handling:** All data processing is based on tables which are similar to the reports and papers used in normal office work. Tables are displayed flat (two dimensionally), but they can represent three (or more) dimensional data. For example, some sales data have three dimensions: time (monthly), region names, and sales values of products. These data can be displayed in two dimensions (shown in Figure 2) by introducing the *group* concept. *STEREO* is a group name in Figure 2.

(b) **No need to learn and memorize command syntax:** As shown in Figure 6, the prompting commands free end users from learning and memorizing command syntax.

(c) **Easy to decide what to do next:** As shown in Figure 7, end users can answer system queries while viewing a current table. Therefore, it is easy for them to decide what to do next.

(d) **Easy to draw business graphics:** The best way to recognize the meaning of data is to represent it graphically. EXCEED permits end users to easily create bar charts or pie charts from their table data.

(e) **Elimination of superfluous typing:** In general, we most frequently refer to our most recent information. Recognizing this human behavior, EXCEED permits users to press the ENTER or RETURN key to indicate that the object most recently referenced is the answer to a system query. Also, a command or object name can be abbreviated to its first few letters as long as the abbreviation uniquely identifies it. The figures in "Session Example" show examples of reducing user typing and abbreviating command and object names. This simplification is very useful for Japanese users, as they are not very familiar with typewriters, and their typing speed is slow.

(g) **Help facility:** As the applicability of a program becomes wider, its users must learn more rules. But end users do not use EXCEED all day long nor every day, so they cannot afford to memorize its detailed rules. And they often forget object names which were created several minutes earlier. To alleviate these problems, EXCEED has a help facility. As shown in Figure 12, if end users press the ? key in response to a system query, EXCEED displays a reminder message listing all possible correct replies.

Figure 12

Example of the Help Facility

```

EXCEED S1-00          LINE .652 COL .1      SALES      82-7-5
ENTER COMMAND: TACOPY
COPY TO DESK OR CABINET? (D/C) : C
ENTER CABINET NAME : ?
ENTER LIBRARY NUMBER AND CABINET NAME, IF YOU ACCESS OTHER USERS' CABINET.
YOUR OWN CABINET(S) ARE FOLLOWING.
AS1      TESTCAB      CAB1      CAB2
ENTER CABINET NAME : CAB1
ENTER TABLE NAME(S) : ?
      PAPER(S) ON DESK ARE FOLLOWING.
+-----+-----+-----+-----+-----+
| NO | DOCUMENT | TABLE | SHEET | CHART |
+-----+-----+-----+-----+-----+
| 1 | | PERSONNEL | S1 | TABLE |
| 2 | | SALES | TESTS | SAMPLEG |
| 3 | | SALES1 | SAMPLESHT | |
| 4 | | URIAGE | | |
+-----+-----+-----+-----+-----+
| CURRENT | | SALES | SAMPLESHT | SAMPLEG |
+-----+-----+-----+-----+-----+
ENTER TABLE NAME(S) : URIAGE
'URIAGE' COPIED.

ENTER COMMAND: .....
.....

```

(2) Characteristics as an application program development support system

(a) **High productivity of application programs:** An application program can be created as a user command consisting of EXCEED basic commands and APL expressions. Typical user commands have thirty to one hundred lines. Since the function of every EXCEED basic command is equivalent to a COBOL program of several hundred lines, and since basic commands cover a wide range of data processing facilities, the productivity of application programs is extremely high.

(b) **Wide range of applicability:** Since a user command is executed as an APL user-defined function, any APL expression can be embedded into a user command so long as it does not destroy the EXCEED environment. Typical APL expressions in a user command are branch, conditional branch, and assignment. If more complex functions are required for the command, higher level APL primitives or operators can be used. To support this facility, EXCEED provides two commands: one to create an EXCEED table from APL variables, and the other to decompose an EXCEED table into APL variables.

EXCEED has been implemented with the Hitachi APL system. This system has facilities to access data in OS files (SAM, DAM, and VSAM), or data in large-scale data base systems, and to communicate with assembler, COBOL, or FORTRAN programs through an interlanguage facility. With these facilities, EXCEED user com-

mands can solve problems which cannot be solved with a combination of EXCEED basic commands.

4. SESSION EXAMPLE

A sample EXCEED session in which prompting commands are used is illustrated in Figures 13-22. The session involves: 1) getting 1981 sales data for stereo players in the Kanto region from *SALES* files and creating a *SALES* table; 2) calculating growth ratios (percentages) for preceding months; 3) drawing sales values as a bar chart with the growth ratios as a straight line graph having the same x-axis.

Figure 13

Creating the Table

(Dialogue with the *TACREATE* Command)

```
EXCEED S1-00          LINE ..22 COL ..1      SALES      82-6-30

ENTER COMMAND: TACREATE
FROM WHICH ? RECORDFILE(R),TERMINAL(T) (R/T): R
ENTER FILE NAME: SALES
CONSTRAINTS: PRODUCT EQ 'STEREO' AND REGION EQ 'KANTO'
              1 FOUND.
ENTER TIME FRAME: 1981 1 TO 1981 12
DISPLAY? (Y/N): N
ENTER TABLE NAME:
ENTER TABLE LABEL:
ENTER TRANSFORMATION FIELD NAMES:
ENTER DATE FIELD NAME: MONTH
OK TO CREATE? (Y/N): Y
'SALES' CREATED.
CONSTRAINTS: STOP

ENTER COMMAND: .....
.....
```

Upon completion of the dialogue, the *SALES* table shown in Figure 14 is created.

Figure 14

Result of TACREATE

EXCEED S1-00	SALES	82-6-30
	STEREO	
MONTH	KANTO(6)	
1981/01	61.7	
1981/02	59.1	
1981/03	61.8	
1981/04	60.3	
1981/05	61.8	
1981/06	62.4	
1981/07	63.9	
1981/08	65.1	
1981/09	67.2	
1981/10	62.9	
1981/11	65.4	
1981/12	70.2	
ENTER COMMAND:		
.....		

Figure 15

Calculation of Growth Ratios

(Dialogue with the *CALCULATE* Command)

Result of *CALCULATE*

EXCEED S1-00	LINE..85	COL ..1	SALES	82-6-30
--------------	----------	---------	-------	---------

ENTER COMMAND: CALC
ENTER TABLE NAME:
ENTER START TIME FOR CALCULATION:
ENTER END TIME FOR CALCULATION:
ENTER CONDITION FOR CALCULATION:
ENTER EXPRESSION: *RATIO=ROUND(PCHANGE(KANTO,1))*
ENTER RESULT TABLE NAME:

ENTER COMMAND:
.....

Upon completion of the dialogue, the *SALES* table is changed as shown in Figure 16.

Figure 16

Result of *CALCULATE*

EXCEED S1-00	SALES	82-6-30
--------------	-------	---------

MONTH	STEREO KANTO(6)	RATIO
1981/01	61.7	
1981/02	59.1	-4
1981/03	61.8	5
1981/04	60.3	-2
1981/05	61.8	2
1981/06	62.4	1
1981/07	63.9	2
1981/08	65.1	2
1981/09	67.2	3
1981/10	62.9	-6
1981/11	65.4	4
1981/12	70.2	7

ENTER COMMAND:
.....

Figure 17

Creation of Chart Definition

(Dialogue with the *CHCREATE* and *CHDEFINE* Commands)

```
EXCEED S1-00          LINE .106 COL .1  SALES      82-6-30

ENTER COMMAND: CHCREATE
ENTER CHART DEF. NAME: SALESGR
CHART DEF('SALESGR') CREATED.
ENTER COMMAND: CHDEFINE
ENTER TABLE OR GRAPH? (T/G): G
FOLLOWING GROUPS OF OPTIONS ARE AVAILABLE.
  1. LABEL
  2. GROUP DEFINITION
  3. LINE(INCLUDING GRAPH TYPE)
  4. AXIS
  5. GRAPH DECORATION
  6. OPTIONS PECULIAR TO NO-TIMESERIES DATA
ENTER NO.: 2
ENTER GROUP,MULTI,OR PART? (G/M/P): G
ENTER GROUP NUMBER: 1
ENTER LINE NUMBER: 1
ENTER GROUP NUMBER: 2
ENTER LINE NUMBER: 2
ENTER GROUP NUMBER: STOP
ENTER GROUP,MULTI,OR PART? (G/M/P): M

ENTER COMMAND: .....
.....
```

Figure 18

Setting Chart Definition Options

(Dialogue with the CHDEFINE Command)

```
EXCEED S1-00          LINE .127 COL ..1    SALES      82-6-30

ENTER RIGHT-HAND Y AXIS OR LEFT-HAND Y AXIS (R/L): L
ENTER GROUP NUMBER: 1
ENTER RIGHT-HAND Y AXIS OR LEFT-HAND Y AXIS (R/L): R
ENTER GROUP NUMBER: 2
ENTER RIGHT-HAND Y AXIS OR LEFT-HAND Y AXIS (R/L): STOP
ENTER GROUP,MULTI,OR PART? (G/M/P): STOP
FOLLOWING GROUPS OF OPTIONS ARE AVAILABLE.
  1. LABEL
  2. GROUP DEFINITION
  3. LINE(INCLUDING GRAPH TYPE)
  4. AXIS
  5. GRAPH DECORATION
  6. OPTIONS PECULIAR TO NO-TIMESERIES DATA
ENTER NO.: 3
FOLLOWING OPTIONS ARE AVAILABLE.
  COLOUR  STYLE  CHARS  TYPE  ASSIGN
ENTER OPTION: TYPE
AVAILABLE GRAPH TYPES ARE AS FOLLOWS.
  TBL STR CRV BOX BAR
  HI1 HI2 HI3 HI4 HI5
  MBR SHI HLC SMB GMB

  _____
ENTER COMMAND: .....
.....
```

Figure 19

Setting Chart Definition Options

(Further Dialogue with the CHDEFINE Command)

EXCEED S1-00	LINE 148	COL 1	SALES	82-6-30
ENTER GRAPH TYPE: BAR STR				
AVAILABLE GRAPH TYPES ARE AS FOLLOWS.				
TBL STR CRV BOX BAR				
HI1 HI2 HI3 HI4 HI5				
MBR SHI HLC SMB GMB				
ENTER GRAPH TYPE: STOP				
FOLLOWING OPTIONS ARE AVAILABLE.				
COLOUR STYLE CHARS TYPE ASSIGN				
ENTER OPTION: STOP				
FOLLOWING GROUPS OF OPTIONS ARE AVAILABLE.				
1. LABEL				
2. GROUP DEFINITION				
3. LINE(INCLUDING GRAPH TYPE)				
4. AXIS				
5. GRAPH DECORATION				
6. OPTIONS PECULIAR TO NO-TIMESERIES DATA				
ENTER NO.: STOP				
ENTER TABLE OR GRAPH? (T/G): STOP				
ENTER COMMAND:				
.....				

Figure 20

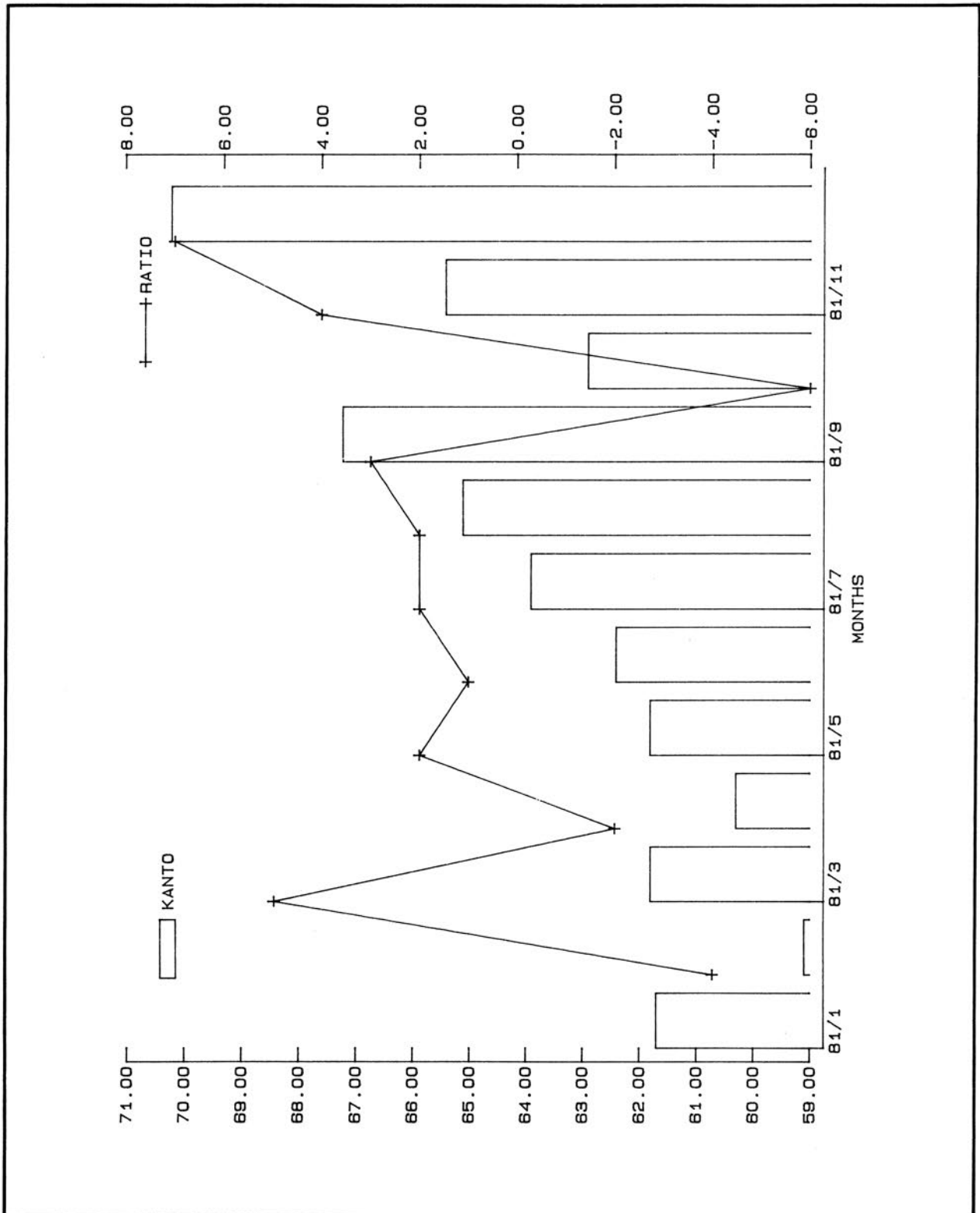
Drawing Business Graphics

(Dialogue with the TADISPLAY Command)

EXCEED S1-00	LINE 22	COL 1	SALES	82-6-30
ENTER COMMAND: TADISP				
ENTER SHEET DEF. NAME: SCREEN				
ENTER CHART AREA AS FOLLOWS.				
DIRECT DEFINITION AS D,BY CHART AREA NAME AS N,FULL SHEET AREA AS F(D/N/F): F				
ENTER CHART DEF. NAME:				
ENTER TABLE NAME:				
ENTER FIELD NAMES TO BE DISPLAYED AS FOLLOWS.				
ALL FIELDS AS A,CONSECUTIVE FIELDS AS C,INDIVIDUAL FIELD AS I(A/C/I): A				
MORE DATA TO BE DISPLAYED?(Y/N): N				
ENTER COMMAND:				
.....				

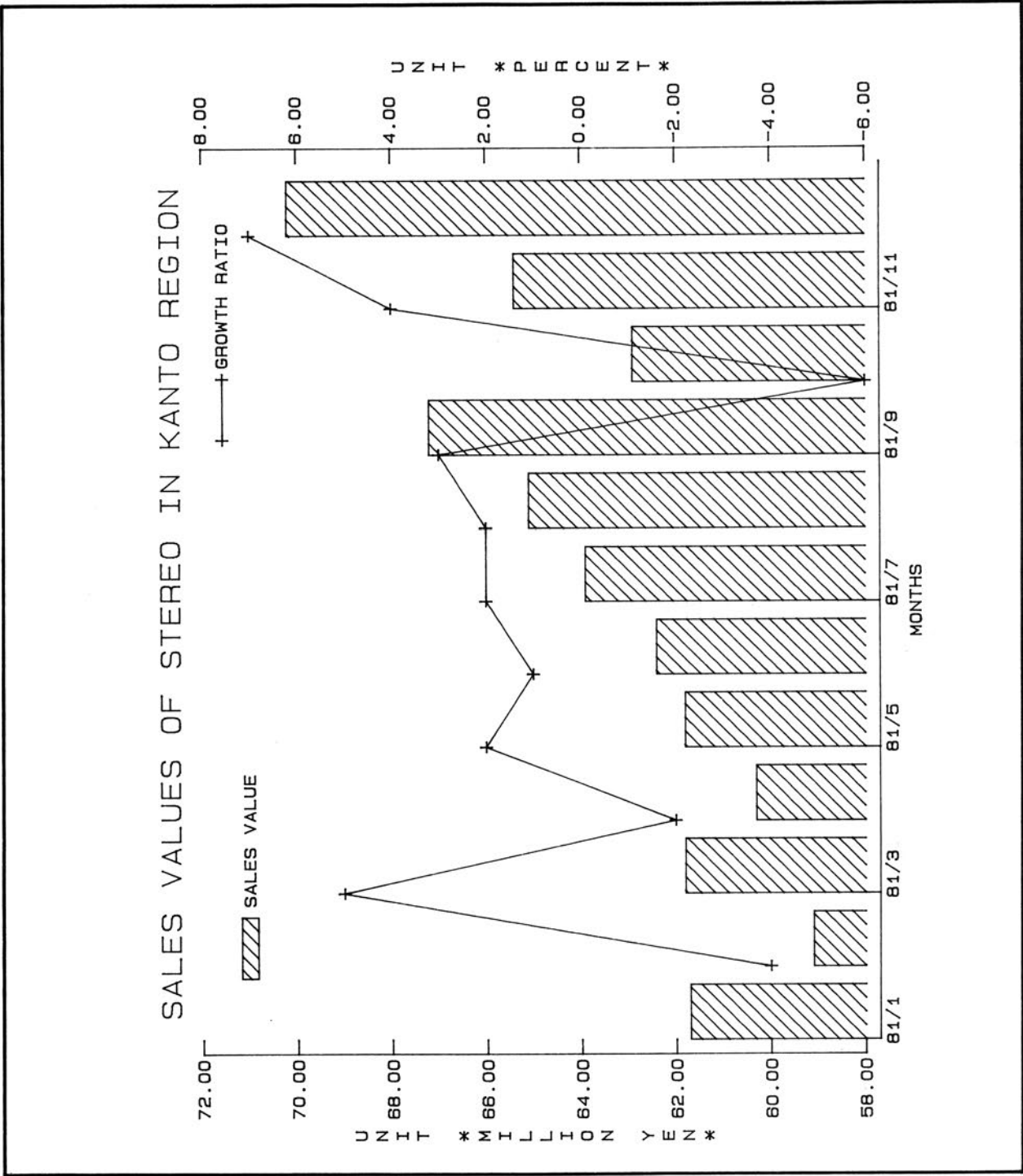
Figure 21

(Graphics Generated by the *TADISPLAY* Command)



If the *TITLE*, *SHADING*, and *Y-AXIS LABELS* options are defined with the *CHDEFINE* command, the more detailed graphic shown in Figure 22 can be displayed.

Figure 22
Detailed Graphics of Figure 20



5. SYSTEM STRUCTURE

In a narrow sense, the EXCEED system consists of the following five components.

- 1) *System manager*: manages user profile, terminal sessions, scroll and command files, and interprets user commands.
- 2) *Record file manager*: manages file space and access privileges, and controls record transfers between record files and the desk (MABRA's and MAGIC's know-how is fully utilized in the implementation of the record file manager).
- 3) *Cabinet and desk manager*: manages cabinet space, desk space and access privileges, and controls data transfer between cabinets and the desk.
- 4) *Data processor and analyzer*: manipulates, calculates, and analyzes table data on the desk.
- 5) *Graphics and report manager*: manages sheet and chart definitions and documents on the desk.

There are several APL programs which support EXCEED. The general structure of the EXCEED system, including those programs, is shown in Figure 23.

APL/BGF is the Hitachi version of SUPERPLOT. A major additional function of APL/BGF is to draw tabular reports. When the graphics and report manager gets an EXCEED command, it generates APL/BGF parameters and calls APL/BGF.

APL/GRAPH is the Hitachi version of SAGA. The major changes to SAGA include the support of the Hitachi T-560/20 color graphic/Kanji terminal, an enhancement of character fields, and a filing facility for terminal screen images to allow scrolling. All terminal input and output is done through APL/GRAPH.

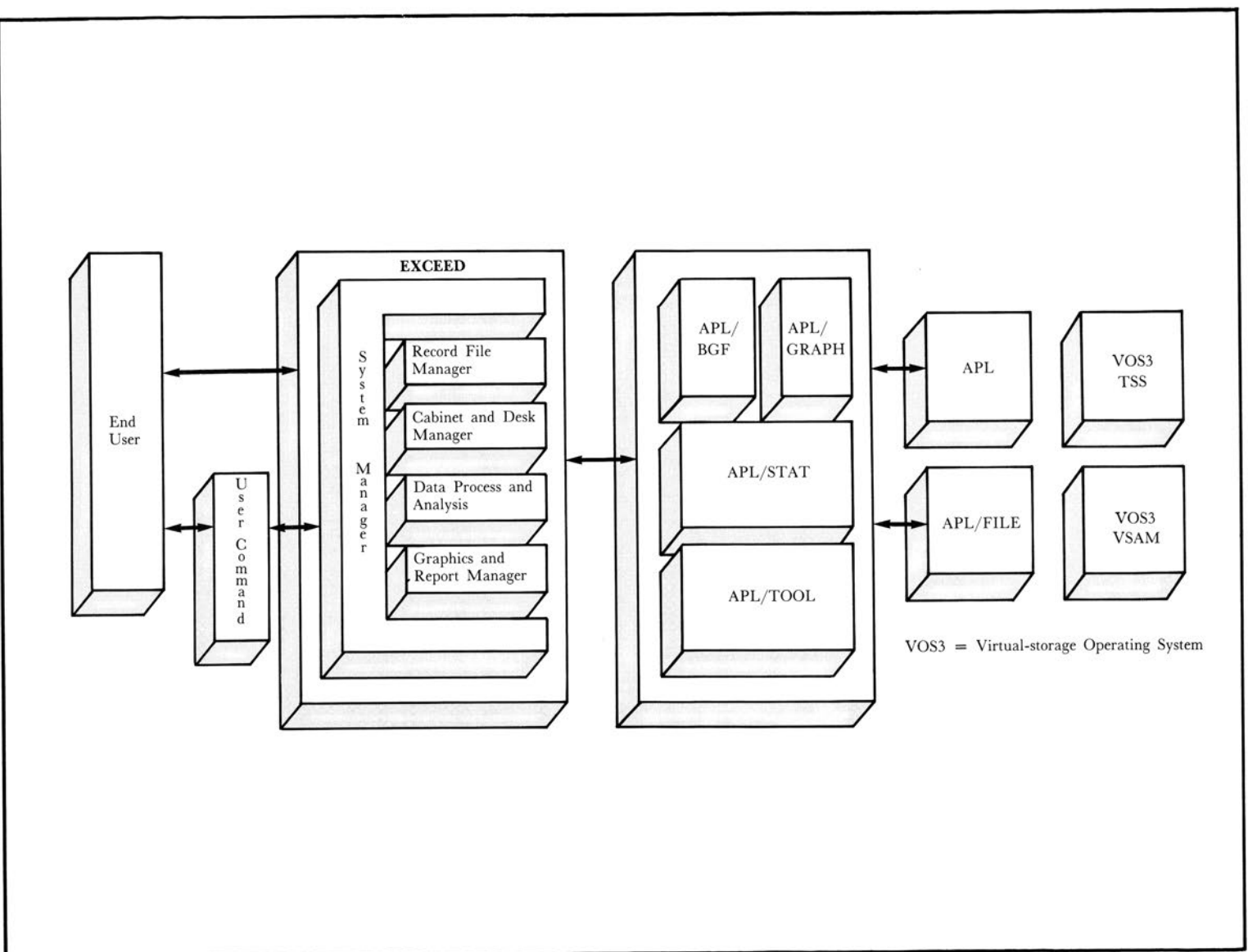
APL/STAT is the Hitachi version of STATISTICAL ANALYSIS LIBRARIES. A data analysis command such as *DASTAT* is implemented by APL/STAT.

APL/TOOL provides the facility for editing APL user-defined functions in full screen mode, and it is utilized to implement the editing facility for EXCEED user commands.

The APL functions used to access APL/FILE are fully compatible with the system functions used to access the SHARP APL File System. All EXCEED data is stored in APL/FILE files. The operating system file structure used by APL/FILE is key-sequenced VSAM.

Hitachi APL is almost compatible with SHARP APL. Prior to the development of EXCEED, package and event trapping facilities were added to Hitachi APL.

Figure 23
Program Structure of EXCEED



6. ADAPTATION TO JAPANESE LANGUAGE

To make the dialogue with EXCEED more natural for Japanese end users, it had to be based on the Japanese language. Fortunately, the Hitachi T-560/20 terminal has a facility to generate and display Japanese characters on the screen. The EXCEED system messages and prompts are displayed with Japanese characters using this facility. A consideration in the development of EXCEED was whether or not terminal input should be in Japanese.

Japanese characters consist of Kanji (Chinese) characters, and Hiragana and Katakana characters which are peculiar to Japanese. Compared to English, Japanese sentences use many different characters. More than five thousand characters are regularly used in Japanese sentences. Therefore, a relatively large keyboard is required for the direct input of Japanese characters. This type of keyboard is not so easy for end users to operate.

Since all Japanese take a three-year English course as part of their compulsory education, and more than half also take an English course in high school, it was reasonable to assume that most Japanese users would know at least plain English words and their syntax. Thus, to avoid requiring a Japanese keyboard for interaction at a terminal, I decided to represent EXCEED commands with English words, and each object name with alphanumeric characters. Figure 24 shows the Japanese version of the *TACREATE* dialogue illustrated in Figure 13.

Figure 24

Japanese Dialogue with *TACREATE*

EXCEED S1-00	LINE 106 COL 1	SALES	82-6-30
--------------	----------------	-------	---------

コマンドを入力して下さい:TACREATE
レコードファイル(R)、端末(T)のどちらから作りますか？(R/T): R
ファイル名を入力して下さい: SALES
検索条件を入力して下さい: PRODUCT EQ 'STEREO' AND REGION EQ 'KANTO'
1件見つかりました
期間を入力して下さい: 1981 1 TO 1981 12
表示しますか？(Y/N): N
テーブル名を入力して下さい:
テーブル表示名を入力して下さい:
フィールド名を階層の順に入力して下さい:
日付フィールド名を入力して下さい: MONTH
作成して良いですか？(Y/N): Y
'SALES' を作成しました
検索条件を入力して下さい: STOP

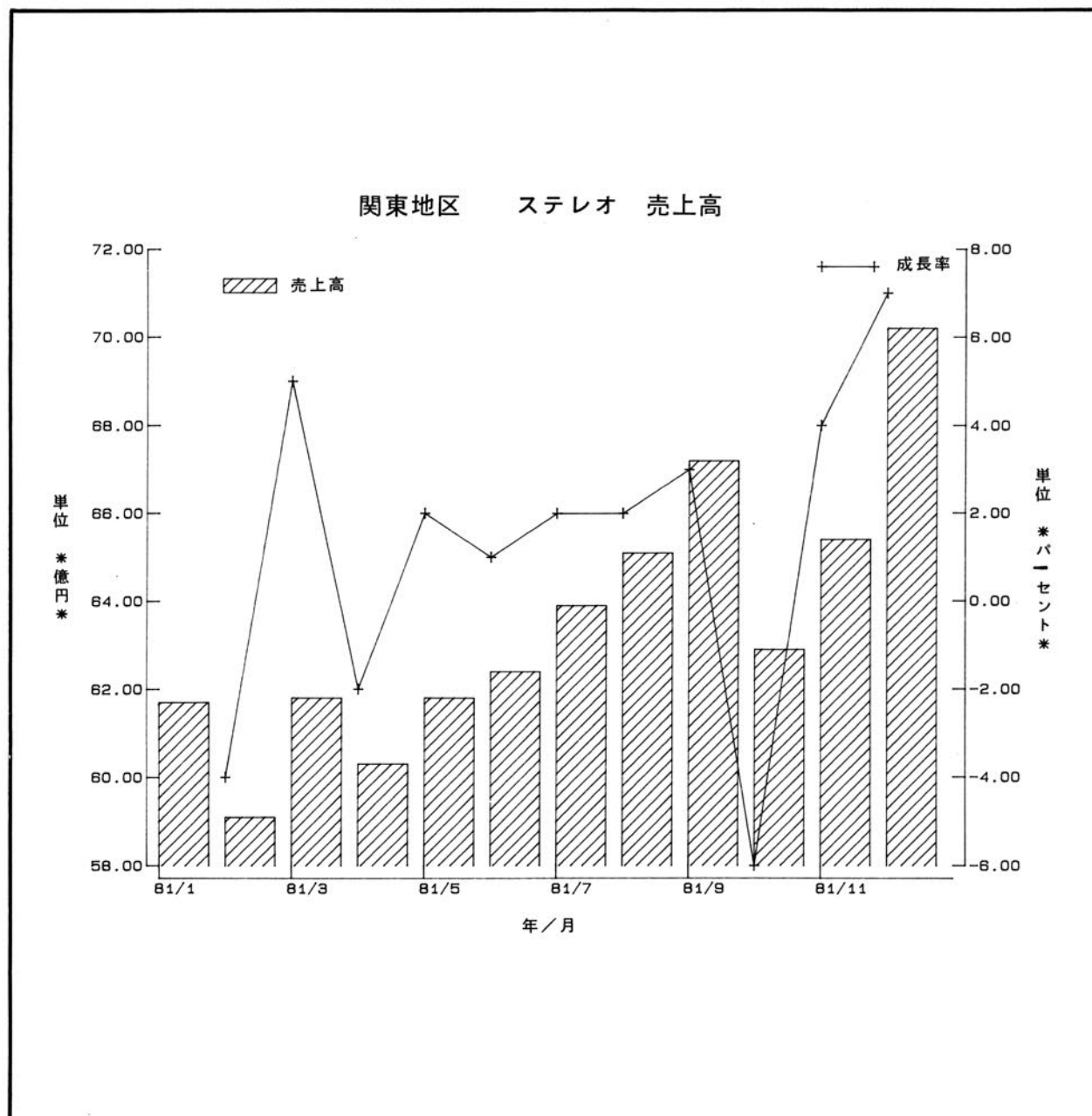
コマンドを入力して下さい:

However, since the final output produced by EXCEED is referred to by Japanese users, it should contain Japanese characters. To accomplish this, every object has two names. One is used to designate an object referred to in an EXCEED dialogue and

the other is used to represent it in a final result such as the output produced by the *TADISPLAY* (display table data), *DOCREATE* (create document), and *DODISPLAY* (display document) commands. For example, a table has two names, a table name and a table label, and every field in a table has two names, a field name and a field label. Table labels and field labels are used by the *TADISPLAY*, *DOCREATE*, and *DODIPSLAY* commands. Figure 25 shows the Japanese version of Figure 22 produced by the *TADISPLAY* command.

Figure 25

Japanese Version of Figure 22



Users also need to enter Kanji words from terminals lacking Japanese keyboards. Almost all terminals used in Japan have Katakana characters as well as alphanumeric characters. Every Kanji word can be represented by Katakana characters which are phonetic. However, a Katakana word may correspond to several Kanji words. For example, 計 which means “total” is pronounced ケイ (kei), but 軽 which means “light” or “not heavy” and 形 which means “shape” are also pronounced ケイ. To solve this problem, a technique which converts a Katakana word to an appropriate Kanji word—with user interaction—was developed. With this interactive technique, table labels and field labels can be represented with Kanji characters. Figures 26, 27, 28, and 29 show the steps in this conversion.


Figure 26

First Step of Code Conversion

EXCEED S1-00
LINE 22 COL 1
82-7-5

```

ENTER COMMAND: TACREATE
FROM WHICH ? RECORDFILE(R),TERMINAL(T) (R/T): T
GENERAL TYPE(G) OR TIMESERIES TYPE(T)? (G/T): G
ENTER TABLE NAME: T01
ENTER TABLE LABEL: ケイ
    
```



Enter table name with Katakana code

Direct to change to Kanji code

↓

```

LABEL TYPE IS CHARACTER OR KANJI? (C/K): K.....
.....
    
```

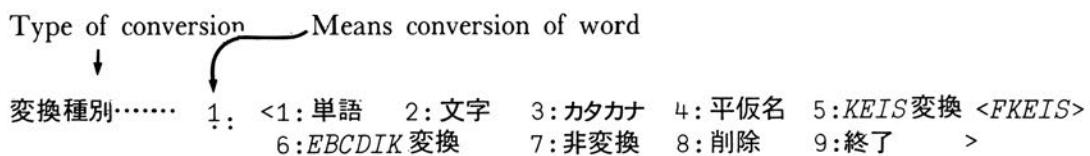


Figure 28

Third Step of Conversion

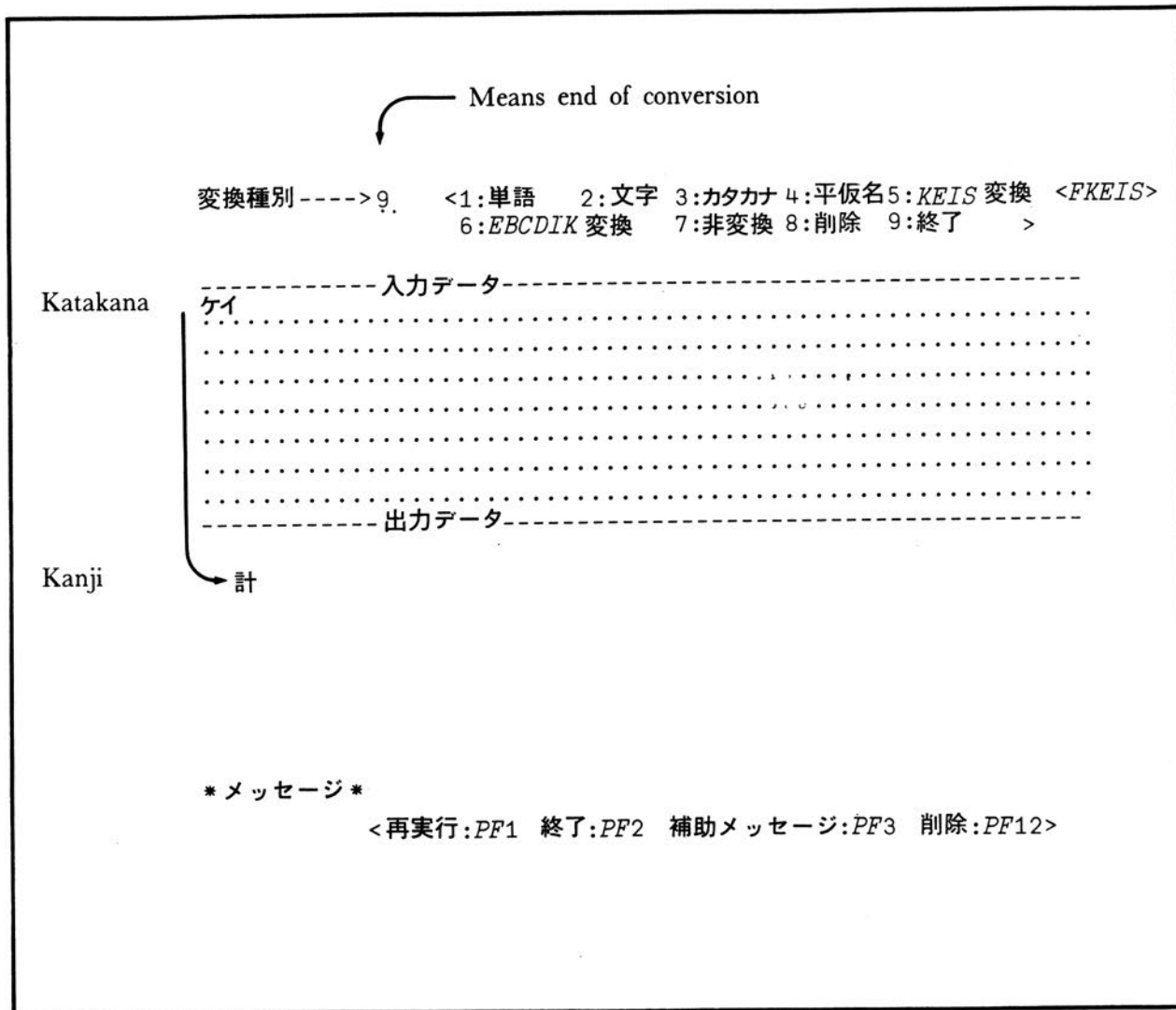
番号' -----> 1 ← Specify appropriate number

ページ 1 <FKEIS>

1 計	}	Kanji words which pronounce as 'ケイ' (kei)
2 軽		
3 京		
4 競		
5 境		
6 係		
7 傾		
8		
9		
10		
11 型	}	
12 契		
13 形		
14 径		
15		
16		
17		
18		
19 携		
20		

< 改ページ:PF3 変換中止:PF4>

Figure 29
Last Step of Conversion



7. ACKNOWLEDGMENTS

We would like to acknowledge the fruitful advice and constructive criticism of Mr. Joey K. Tuttle of I.P. Sharp Associates, Mr. Takeo Kaku of Nippon Steel Corp., and Dr. Fumihiko Mori and Mr. Hiroshi Tsuji of the System Development Laboratory of Hitachi, Ltd.

APL BEYOND THE ACTUARIAL DEPARTMENT

Tim P. Fitzpatrick
Financial Life Assurance Company of Canada
Edmonton, Alberta

Introduction

From this Users Meeting it can be learned that most APL users are not professional programmers. Rather, most have another major profession and use APL as a tool to enhance their own work. Many more than likely became involved with APL when they were searching for a faster solution to their problems than the traditional EDP route.

APL in insurance companies

APL probably made its first inroad to each insurance company when an actuarial student in the early 1970's pointed out that he could do a required program himself rather than delegate it to the data processing department. This is how it happened for me when I worked at the Dominion Life Assurance Company in Waterloo, Ontario. Our first programs were in FORTRAN. Programming was done directly in the actuarial department over the next two years. The most elaborate FORTRAN program written was to perform a gross premium valuation on a major block of business that was to be sold. After that project, we felt that we needed a more flexible language and on-line programming capability. It was at this point that we decided to begin APL time-sharing with I.P. Sharp Associates.

Traditional actuarial applications

I was one of four actuarial students that took a crash course in APL. From there, the use of APL for traditional actuarial applications expanded quickly. APL for actuaries is rather a natural. Sometimes, it was rumoured that APL stood for "Actuarial Programming Language". The use of vectors and matrices directly parallels actuarial theory. In addition, most actuaries seem to have an aptitude for programming, especially in APL.

The availability of ACTPAK was and still is a big help. It is very useful for quick, one shot applications. The APL terminal then becomes simply a powerful calculator that replaces worksheets and commutation columns. I.P. Sharp maintains an extensive and up-to-date data base of major actuarial tables. Even if ACTPAK is not used directly, the availability of the tables make things simpler.

At Dominion, we used APL for all of the traditional applications. The most obvious is pricing. Asset share systems are extremely simple to write in APL. Being interactive, the product development actuary is able to receive immediate feedback from changes in assumptions, allowing him to try many more combinations. This leads to faster, more economical and more scientific product development. After the pricing is completed, the production of the rates and values is again very easy. At Dominion Life, we produced camera ready printouts that, when put together with an overlay, combined to make instant rate pages. This cut down typesetting costs significantly and improved turnaround. Rate cards and pages for the policy contracts are produced similarly. Magnetic tapes in the required format were created to update various inhouse rate files. APL at Dominion Life greatly reduced costs while improving accuracy and turnaround for product development.

We also used APL to create a model office. Annually, we performed a five year projection for the company broken down into eight lines of business. The Controller and the Actuary were then able to try many "what if" combinations. The result was a much more consistent five year projection.

Another major use of APL in the Actuarial Department was for valuation. At first, it was used simply as a powerful calculator for the more insignificant lines of business. This changed when the valuation rules changed in 1978. Within a matter of months, one actuarial student was able to produce a magnetic tape that contained tables of factors using the new valuation method. The tape was formatted so that it would fit into our existing in-house valuation system. As a result Dominion was able to meet the changed valuation methods with relative ease. Dominion Life's commitment to the annuity market encouraged the development of an elaborate annuity quotation system. Since actuaries tended to spread out throughout the company and were routinely rotated, APL spread to the Group and Pension Departments. I was assigned to work with another actuary in the Marketing Department to redesign the compensation system for Dominion Life. Again, APL was the primary workhorse. Modelling programs were developed to compare different compensation systems with other companies' agent contracts. Nine other companies' contracts were converted into APL code. When an agent profile was entered, all ten companies were ranked based on different criteria. As a final step, a data base was created containing information on the existing field force. The effects of the different compensation systems were tested on an individual basis, allowing us to choose the most appropriate for Dominion Life.

Other applications

After only a few years, APL had become quite a fixture at Dominion Life. To this point, however, it had only been used to help the actuaries. The data processing people admitted that APL had some use for one shot applications and thereby relieved them of a lot of nuisance work. But they steadfastly maintained that APL should never be used for production work. Being in the Marketing Department already, I was empathetic with the desire of the marketing people to provide elaborate computerized sales illustrations for the field force. The best the Data Processing department could offer was a feasibility study with a timeframe of one year. Marketing was all too familiar with commitments like this being changed due to higher priority items. It caused quite a political uproar when I suggested that it could be done in a matter of few months. The traditionalists have always argued that APL should not be used as a production language since it makes large demands on machine resources. Production programs should be able to be written in APL that perform in a cost effective way for most applications. Certainly if you fail to take advantage of the data structures and "non-

looping" facilities of APL any comparison becomes unfair. Another standard argument is that APL programs are more difficult to read and therefore to maintain. Even if both these points were valid, there is even a stronger counter-argument. Although no one knows the exact factor, it is clear that it takes far less time to program in APL than in traditional languages. The ratio of 10 to 1 is commonly used. Every year, machine costs are dropping and people costs are skyrocketing. In addition, the life span of most systems is decreasing rapidly with the changing environment. Therefore, the savings in development costs far outweigh any extra machine or maintenance expenses. But a by far most compelling reason is that you can have the system today instead of in two years.

I am convinced that traditional EDP types dislike APL because it threatens their very existence. For a decade, the high personnel requirements of computer technology have created a shortage of skilled specialists and driven up salaries. For years, they've been able to convince management to make huge expenditures on data processing. The fact is, if a ten to one ratio is valid, then 90% of analysts and programmers would be out of work. In addition, programming is the only skill most of them have. APL tends to lend itself to decentralization by placing the responsibility of system designs directly with the users. As I originally stated, most APL users have another profession. So the decision to write production programs in APL became very political, yet the decision was made. The possibility remained, after the design and implementation of the APL version of a system, for the D.P. department to produce an equivalent system with lower running costs. This process would provide the desired system when it was needed and would reduce substantially design and implementation costs.

The sales illustration system was up and running in a matter of months and soon provided 50 illustrations a day. The average time-sharing cost for an illustration was \$2.50. The system was designed to keep these costs to a bare minimum. All rates and values were calculated by formula, not by table look-ups, which resulted in many of the advantages described earlier. For instance, once the system was up and running, it was not hard to convince management to conduct an experiment with terminals in branch offices. We are able to offer sales illustration systems along with sister systems for annuity quotations. As well, electronic mail offered immediate savings over existing telex facilities. The final outcome was the bringing of APL in-house in a major scheme to provide on-line terminals in all branches. Dominion Life made a major product change to Universal Life, which made the sales illustration system outdated. It is a good example of a system having a life span of only a few years. If we had used traditional data processing methods this system would have been outdated before it had even been implemented. I think there is a lesson to be learned here.

Financial Life - A new challenge

A year ago I joined a small company called Financial Life that was interested in expansion. Financial Life has a Head Office staff of approximately 30 people with a budget of approximately \$1 million. The company employs a data processing staff consisting of one analyst and has no in-house computer system. All computing at that time made use of a time-sharing service located in Edmonton. To the analyst and Accounting Department's credit, the company already had a commission and billing system that worked very well.

The desire for fast growth and the company's small size had two implications:

- 1) Innovative products in today's marketplace need technological backup to compete.
- 2) A major component of pricing of life insurance is expenses. My goal was to keep per unit cost not just equal to, but less than the more established companies.

For Financial Life, the advantages of APL were enormous. We could not afford the time nor expense of traditional computing. In a small company, you have to wear a lot of hats that means as well as developing the products, I have to design and write the backup administration system. In order to use low per unit expense assumptions in my pricing, I am able to use APL to give us a market edge. Of course, APL was used extensively in the design and pricing of the new products. The first major system I wrote is designed to administer an accumulation annuity. Within a month, a system was put in place that provided:

- 1) Printed face pages
- 2) On-line status enquiries
- 3) Commission statements
- 4) Monthly policyowner statements
- 5) Reserves
- 6) Accounting reports

The product has averaged approximately 100 applications and \$300,000 of premium per month since introduction. Since this system is fully automated, administration of the product requires less than 50 man hours per month.

Work on an elaborate life insurance processing system is underway. When complete, the system will provide:

- 1) Face pages
- 2) Sales illustrations
- 3) Status enquiries
- 4) Marketing reports
- 5) Reinsurance cessions

Afterwards, I plan to install terminals directly in our underwriter's offices. Every action that occurs during the underwriting of an application will be recorded. Early in the spring we will then install terminals in our Regional Brokerage Offices. Besides electronic mail and all of the above features, the offices will be able to track the progress of an application through to underwriting. They can make enquiries and produce listings, such as all transactions that occurred pertinent to their office in the previous day. Or they could look at the history of the underwriting of a particular policy and issue a message via electronic mail to one of the underwriters if necessary.

Summary

APL at Financial Life is still in its early stages. So far it has helped:

- 1) Develop a product
- 2) Service a product
- 3) Competitively price a product

New APL applications, relevant to the work being done at Financial Life are constantly appearing. It is my hope that APL will continue to help Financial Life maintain a competitive edge.

RUNNING A MONEY MARKET SYSTEM ON SHARP APL

Graham Parker
Assistant Treasurer, Money Market
CRA Ltd.
Melbourne, Australia

To those who have faced the problem of creating a system of operating and management control in an area previously untouched, or handled in a completely different way, this paper will follow a very traditional outline:

- 1) Identification of the problem
- 2) Possible solution
- 3) Outline of how the system works

Identification of the problem

CRA is a very large and complex organisation with assets of \$4.7 billion in 1981, making it a significant mining house by world standards. (See Figure 1)

The company originated in 1905 in the noted silver, lead, zinc deposits at Broken Hill and has subsequently diversified into large scale mining of copper, uranium, coal, iron ore, salt, and now diamonds. The treatment smelting, refining and fabrication of many of those minerals is also undertaken by the company. (See Figure 2 for organisation structure)

Whilst the CRA Group has developed as a diverse range of enterprises, many with significantly different needs, it is felt that some functions are still best centralised, two of which are corporate finance and funds management, as long as recognition can be made of individual company requirements.

Until early 1981 each CRA operating company was primarily responsible for its own cash control, borrowings and investment of surplus funds. Whilst there was a significant co-ordinating influence from the parent company, the emphasis remained on separate operating units.

The decision was taken to set up a new wholly owned subsidiary of CRA, CRA Finance Limited, to act as the "banker" to the entire group. Whilst the concept is delightfully simple the technicalities of operation are rather more complex.

Possible solutions and rationale for decision taken

The first hurdle for CRA Finance was the management of the domestic currency cash surpluses of the group, in itself an operation involving the investment of between A\$200-A\$300 million, some 500 accounts and up to 100 transactions per day.

If only for manpower considerations the project was unmanageable without computerisation. The options ranged from development of our own software to be run on CRA's Univac main frame, through the mini computers to software packages operated on bureau facilities and all the combinations and permutations therein. In the investigation that was undertaken some of the factors subject to consideration were:

- cost (both front end and operating)
- flexibility
- system integrity
- audit control
- reliability
- system support

The decision making process turned out to be one of elimination:

- 1) It was felt to be too expensive and certainly too slow to design our own system.
- 2) Although there were several packages being actively marketed, none were exactly suited to our requirements and the issue became, "how quickly and at what price those packages could be adapted". On this occasion SHARP APL had no real competitor.
- 3) The other main issue was whether we would operate the chosen software "in house" or on the I.P. Sharp bureau. To have operated "in house" would have caused financial commitment to hardware and would have removed us from the mainstream of on-going developments in exchange for possible savings in operating costs. We chose the bureau.

Operation of the Money Market system on SHARP APL

The system is currently designed as a series of separate modules which can be operated independently but which also integrate with a common general ledger. The modules currently being operated are:

- 1) Money Market
- 2) Securities Trading
- 3) International Money Market
- 4) Foreign Exchange
- 5) General Ledger

Within the operating modules it is currently possible to duplicate the system up to tenfold.

Money Market

The Money Market system provides facilities for deposit and advances and the renegotiation of terms on any of those transactions. An essential feature of the system is the on-line screen inquiry facility.

This enables a dealer, by keying an alpha code, to have instant access to a client's financial transactions with the company, both deposits and advances, accrued interest details, account history and monetary limit and exposure details. The response speed and reliability of the screen inquiry system is a vital aspect of the operation.

The whole system is designed to minimise and simplify the flow of paper work and only a minimal number of reports need to be printed daily—one being a “daily action report” which itemises all maturities for the day and overnight transactions, and the other a management report. However, a full suite of reports are available, many of which are printed on a monthly basis.

Similarly, data preparation has been simplified to the extent that all Money Market transactions are recorded by the operator using only one type of voucher. (See Figure 3)

The voucher has been designed to operate not only as a record of a Money Market transaction but also as a cheque requisition, data input document and general authorisation.

We have chosen a mode of operation whereby the Money Market dealers can make inquiries into any account, but because they do not hold the password, they are unable to amend the data base.

Similarly we choose to use a batch update system rather than on-line because it strengthens internal control and simplifies data input control.

Securities Trading module

This is the natural complement to the Money Market module which handles exclusively cash whilst this handles negotiable instruments.

In Australia we have a well developed market for the buying and selling of short term negotiable securities of various types ranging from government-issued treasury notes and bank accepted commercial paper to corporate promissory notes, and the operating concept of the Securities Trading module reflects these various types of instruments. Just as the Money Market system is dominated by an on-line screen inquiry the securities system depends on the register concept.

The hard copy dealer report is a full register of all securities, by type and maturity profile and most importantly provides the dealer with a break even-yield for each parcel of securities as a guide for profitable trading.

The system on the same single voucher concept allows for the purchase, sale, maturity and indorsement of securities together with the calculation of entries relating to profit and loss in trading. (See Figure 4 for security deal voucher)

General ledger

The reader may have noticed in examining the cash deal voucher and Securities Trading voucher that both are designed on the double entry bookkeeping system. This aspect was necessary for CRA Finance, as the company is designed to operate as a profit centre and to report in its own right. The computer accounting module generates accounting data up to the trial balance stage.

The preceding outline summarised the basic software to which CRA Finance was committed for the support of its A\$ cash operations.

However, nothing stands still and with the increasing emphasis on foreign currency transactions we have just recently loaded two further units of software, one handling the international Money Market transactions based in Hong Kong and the other a foreign exchange system.

International Money Market

Australia operates a currency control system whereby foreign currency holdings must be returned to Australia without delay. However, the Australian Reserve Bank is willing to give permission for foreign currency proceeds to be held off shore if there are known and imminent payments which must be undertaken. We have been able without any difficulty to load on the computer system a foreign currency money book which mirrors in all aspects the domestic Money Market system and also integrates with the general ledger, with valuation of foreign currency holdings being effected through a transfer account.

Foreign Exchange system

As a further development of the CRA Finance role as banker to the group we are beginning to account for the group in foreign exchange transactions. The Foreign Exchange module has been designed to account for foreign currency hedging and spot transactions. Given that a currency hedge is a means of eliminating an exchange risk by fixing or guaranteeing a future exchange rate, for a mining company trading internationally this becomes a vital aspect of operations.

The foreign exchange system enables on-line dealer inquiry into any client's exposure and limit and inquiry into the book position in any currency and/or time frame. One further refinement in this system is the capacity for the dealer to immediately enter his deals into the screen inquiry enabling a constant update on position.

However as in the Money Market system this screen update is for the dealers information only and cannot affect the data base which is updated in the same way as with the other modules.

In Australia currency hedge is non-deliverable and as such settlement takes place for the net difference only. As part of the settlement routine the system generates automatic postings for the profit and loss on each deal and a client settlements advice. As with the other modules a full range of reports is generated.

Conclusion

We have found that the software we are operating on SHARP APL has proved to be readily adaptable to our evolving requirements and we look forward to our next phase of development which is likely to be a loan administration module enabling us to track and maintain our growing portfolio of multi currency loan raisings and to simplify much of the complex problem of valuation and exchange fluctuation.

Figure 1

Metals and Mining: Nonferrous Metals, Iron Ore, Etc.

	FY	TOTAL ASSETS 1981 US\$M	TOTAL DEBT 1981 US\$M
Allegheny International	12	1423.7	356.8
Aluminum Co. of America	12	5632.0	1280.0
AMAX	12	5460.1	1528.6
ASARCO	12	2047.4	415.0
Cabot	9	1258.9	300.3
Cleveland-Cliffs Iron	12	615.9	36.4
Commercial Metals	8	243.4	18.6
CRA	12	4717.3	903.9
Gulf Resources & Chemical	12	441.0	238.3
Handy & Harman	12	365.0	170.6
Hanna Mining	12	631.8	76.6
Harsco	12	673.1	128.7
Homestake Mining	12	333.6	0.6
Inco	12	3774.0	1331.3
Kaiser Aluminum & Chemical	12	3611.4	1031.7
Kennametal	6	339.4	60.2
Newmont Mining	12	1843.3	197.5
Phelps Dodge	12	2174.2	678.2
Porter (H.K.)	12	319.3	75.5
Revere Copper & Brass	12	498.8	183.4
Reynolds Metals	12	3332.1	1080.5
UNC Resources	3	395.8	81.0

CRA
LIMITED

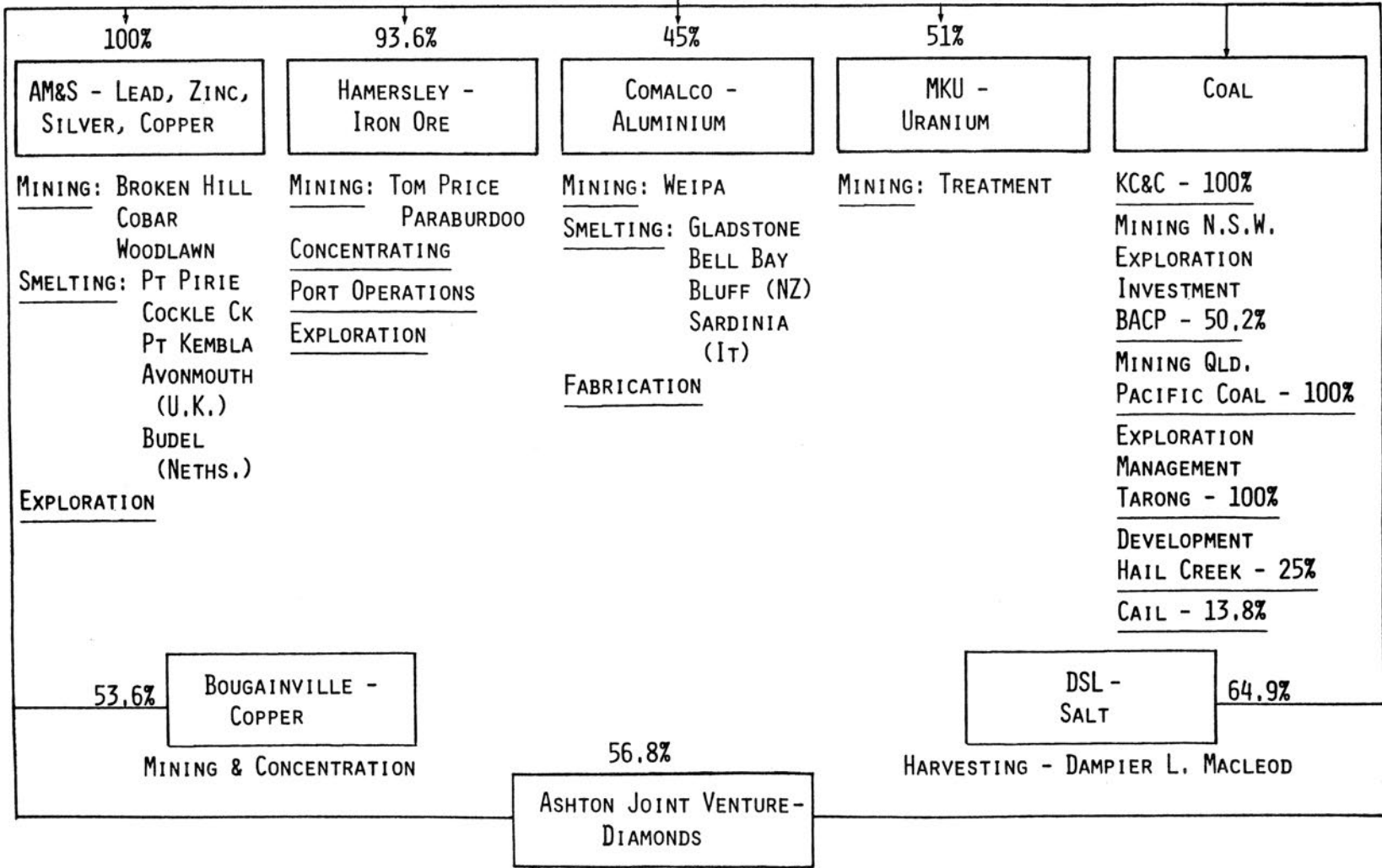


Figure 2

CRA Finance Limited

CASH DEAL VOUCHER

Broker

Client

New
☐

Settlement Date

Deal No.

2859

	Account	LR	Comment	Description	DR	CR	Rate	Term	Term Code
DEPOSIT		10	Secur.	DEPOSIT					
		11	Unsec.	DEPOSIT					
	12600	50	Bank						
REPAY DEPOSIT		10	Sec.Dep.	REPAY					
		11	Uns.Dep.	REPAY					
	12600	50	Bank	CHQ.NO. : : : : : PAYEE					
ADVANCE		12	Secur.	ADVANCE					
		13	Unsec.	ADVANCE					
	12600	50	Bank	CHQ.NO. : : : : : PAYEE					
REPAY ADVANCE		12	Secur.	REPAY					
		13	Unsec.	REPAY					
	12600	50	Bank						
INTEREST		22	Sec.Adv.	INTEREST RECEIVED					
		23	Uns.Adv.	INTEREST RECEIVED					
	12600	50	Bank	INT.REC. CLIENT					
		20	Sec.Dep.	INTEREST PAID					
		21	Uns.Dep.	INTEREST PAID					
	12600	50	Bank	CHQ.NO. : : : : : INT. PAID CLIENT					
SUNDRY		50	Fun.Rec.						
	12600	50	Bank						
		50	Fund.Pd.						
	12600	50	Bank	CHQ.NO. : : : : :					

SETTLEMENT INSTRUCTIONS
☐ Our Bank ☐ Our Office ☐ Our Cheque
☐ Their Bank ☐ Their Office ☐ Bank Cheque

Initials	Dealer	Cash Sheet	Letter Signed	Cheque Signed	Limit Checked	Entered

Figure 3

CRA Finance Limited

SECURITY DEAL VOUCHER

CRA Finance Limited										Client		NEW <input type="checkbox"/>	Settlement Date / /		Deal No 7381		
Details	Account	LR	Drawer	Description	Accept	Endors	Debit Face Value		Credit Face Value		Rate	Matur. Date D M Y			Days	Considerat'n	
PURCHASES				/	/												
				/	/												
				/	/												
				/	/												
				/	/												
				/	/												
Interest Earnable	13100	50															
Amount Paid to Client	12600	50	CHQ.NO			CLIENT											
Bank Charges	15200	50															
MATURITIES				MATURITY	/	/											
				MATURITY	/	/											
				MATURITY	/	/											
				MATURITY	/	/											
				MATURITY	/	/											
				MATURITY	/	/											
Amount Received from Client	12600	50		MATURITY	/	/											
SALES				SOLD	5	PURCH. DEAL	/	/									
				SOLD	6		/	/									
				SOLD	6		/	/									
				SOLD	6		/	/									
				SOLD	6		/	/									
				SOLD	6		/	/									
Interest Written Back on Sale of Bills FV-A	13100	50															
Profit on Sale of Bills A-B (Cr)	20500	50															
Loss on Sale of Bills A-B (Dr)	15900	50															
Amount Received from Client	12600	50															
SALES CALCULAT.	STEPS										SETTLEMENT INSTRUCTIONS <input type="checkbox"/> Our Bank <input type="checkbox"/> Our Office <input type="checkbox"/> Our Cheque <input type="checkbox"/> Their Bnk <input type="checkbox"/> Their Off. <input type="checkbox"/> Bank Cheque						
	1 Calculate Total Value for A & B 2 Post (FV-A) to Interest Written Back on Sale of Bills 3 Post (A-B) to Profit on Sale of Bills IF CREDIT 4 Post (A-B) to Loss on Sale of Bills IF DEBIT						TOTAL DEBIT		TOTAL CREDIT								
	A	FV-A	B	A-B		INITIALS											
	Current Value at Break Even Rate	Interest Written Back on Sale of Bills	Current Value Quoted for Sale	Profit / Loss on Sale of Bills		Dealer		Cash Sheet		Letter Signed		Cheque Signed		Endorsement Signed		Limit Checkd Entered	

Figure 4

APL AS A TOOL FOR FINANCIAL APPLICATIONS

**Denis Lefebvre
Project Manager
Financial Services
Dominion Textile Inc.
Montreal, Quebec**

Introduction

Since it was first introduced at Dominion Textiles Inc. in 1975, SHARP APL has gained increasing acceptance and recognition as a problem solving tool. At present, it is being used by many departments within the organization.

At the end of 1981, a system was developed whereby subsidiaries of Dominion Textile Inc. in Europe, U.S., and Canada could enter financial data via the communications network producing consolidations at various levels up to the final corporate consolidation in Montreal.

The purpose of this paper is to present a) considerations in the development of APL projects, b) the environment in which these are being developed and c) the consolidation and reporting, and budgeting systems which were developed recently. Finally I will describe the control mechanism established to ensure proper use.

Development in an APL environment

The main object in using APL is to provide corporate and divisional controllers' departments with analytical tools designed specifically for generating timely and accurate information required in the planning and decision-making processes. APL is an attractive tool; but we must keep things in perspective, as APL applications are expected to complement our computer requirements. The general criteria for the selection of projects to be developed in an APL environment are:

- application is not well defined
- manual system does not exist
- application is complex
- specifications are evolving continually
- volume of information to be processed is relatively low
- development time is a constraint
- application will be run infrequently: on a monthly, quarterly or yearly basis, or only for the life of the project evaluation

- the project relates to a decision affecting the future

In summary, applications can be developed in a matter of days and the volume of data is limited, which means that the start-up cost of the application is reasonable. This exploratory development highlights the expected benefits. Further enhancements and/or expansion of the data base can then be justified. In addition, this step-by-step approach offers a better chance of success as the user can adjust his needs in relation to the actual results.

How can we develop effective, easy-to-use and useful systems? The best way is through a partnership between users and system people. The user must understand what the computer can do to meet his needs and then justify it from a business perspective. The APL specialist must in turn understand the business problem, show the user what is possible with the computer, and formulate an effective system solution which is understood by the user. This approach works well in our organization as the user's prime interest is in getting the job done and examining the results; as changes are required, they are done quickly and effectively. We provide the user with a set of commands and the user can run as many simulations as he feels necessary to achieve his ends.

Financial applications

Up until now, the presentation has focused on how APL applications are being selected and developed. The following is a review of the consolidation and reporting system, implemented in the fall of 1981, and the profit plan system, implemented in the early spring of 1982.

Consolidation and reporting system

The purpose of this application is to establish a standard computerized data base to enable preparation of management reports and financial statements for each corporate entity, and to facilitate timely reporting at various reporting levels.

How was it achieved?

- Decentralization of data entry at the unit level: this was made possible by using the communications network. Data is entered through a terminal or (in some instances) a Telex machine. This has the advantage of making the data available sooner.
- The system comes with a limited set of available commands. They are set up in a structure which makes it easy to enter data into the system, list it, and produce the final report when data is finalized.
- There are three types of financial year-ends: Canada—July to June, U.S.—June to May, Europe—April to March. To avoid ambiguity, it was decided to use the term "period." Period 1 refers to July in Canada, June in U.S., and April in Europe. This has simplified the file structure lay-out as consolidation is done period by period.
- Standardized forms were set up, to be used by all companies.
- Linking the various units of the reporting structure: this makes each user

knowledgeable and responsible about the status of units to be completed prior to beginning his consolidations. It also helps to pin-point who is holding up the consolidation.

- Node numbering is divided into the following categories:
 - business units
 - consolidated adjustments
 - translation adjustments
 - consolidation groups in the financial reporting structure
 - consolidation groups in the management reporting structure

The intention was to facilitate the understanding of the different nodes (there are over 200 nodes defined).

- Data is entered one period at a time. Verification for consistency of data is maintained by cross-checking specific lines in the forms and subforms.
- A command to list data was set up: this prints the line numbers with the corresponding data but only for those lines (input and calculated) where data is non-zero. Again the system verifies for consistency of data.
- A report is also available which produces results as forms. This is normally done when data is final and a hardcopy is required for reporting purposes.
- Data entry for adjustment nodes is set up on a different basis than units. All journal entries are made following accounting format—debit (input as positive number) to increase cost, expenses and assets, and decrease income, liabilities and shareholders equity; credit (input as negative number) to increase income, liabilities and shareholders' equity, and decrease cost, expenses, and assets. In addition, transaction entries within an adjustment number are grouped by reference number. This has the advantage of identifying the different kinds of adjustment done; for example, elimination of sales between companies in the group, receivables and payables, adjusted inventories at historical rates. Finally, a verification that transactions balance to zero is done.
- The system has been revised to summarize the entries and present the results accordingly. The line numbers can be listed with the corresponding data or all the transactions can be listed by reference number—one adjustment number at a time.
- During the consolidation, the computerized system automatically converts the local currency statements to the currency of the parent corporation. The statements of income and the statement of retained earnings are converted using the average rates. The statements of working capital and statements of financial position are converted using the closing rates. The resulting transaction gain or loss is added to, or subtracted from, the statement of income.
- The same kind of reporting facilities exist for consolidation groups as for units and adjustments (as explained above). In addition, there is a command

that prints reports for groups along with the nodes reporting to the group. Results are in the currency of that group.

- Tables of closing exchange rates are entered in Montreal for all the defined currencies in the system. Average exchange rates are calculated by dividing the sum of the rates for the opening and the closing periods by the number of periods plus 1. This activity is centralized to ensure that proper and consistent exchange rates are used. Translation of statements cannot be done until exchange rates are final.
- Security has been built in so that units cannot be read (for consolidation purposes) by the parent corporation until the units, adjustments, and groups are closed. Then the data for the period becomes available on a read mode only. Beware—as this data refers to actual results, users should not be allowed to change it at will.

Many changes were requested during the initial design and set-up. All were incorporated to provide the user with a better tool to work with. In addition, there are plans to go from quarterly to monthly reporting and to incorporate a forecasting facility.

Profit plan system

Immediately after the implementation of the consolidation and reporting system in the fall of 1981, the profit plan system was redesigned to conform to the same objectives. It was in production by February 1982 for use in Europe, the U.S. and Canada.

Forms were revised to conform to the other system. Historical information was required and, therefore, two forms were prepared, one for yearly data (3 years historical data plus plan and estimate for the current year), and the other for planning quarterly data for next year. A different file was required, as planning data is on a quarter by quarter basis while the consolidation and reporting system was on a cumulative basis. Other than that, the system was set up to behave the same way as the reporting system and with the same commands. At the end of the profit plan activity, quarterly data for the planned year is transferred into the consolidation and reporting system file.

Conclusion

The applications just described represent a portion of the activities using SHARP APL. As usage increased, it became important to know if spending exceeded budgeted amounts, or if spending made it justifiable to review the current activity for that application. Using programs available in the workspace 1 *USAGE*, three different types of reports were prepared:

- 1) A report showing variance from budget, by department.
- 2) A detailed report of connect time, CPU , storage, and character charges, along with some ratios such as percentage of batch processing versus on-line, by application, with a summary by department.
- 3) Finally a report of year-to-date charges by application and by department.

These reports are produced at the beginning of every month and a copy is distributed to the department heads.

USING DEFINED OPERATORS

Jon McGrew
IBM Corporation
Kingston, New York

ABSTRACT

Most of us are used to defining our own APL *functions*, and many of us have followed the various Functions-versus-Operators discussions over the last few years. This paper describes a recent implementation of user-defined *operators*, explains the mechanics of defining them and discusses some applications of them.

IBM's recently-released APL2 now allows operators to be user-defined. By defining your own operators in the same way that you define your own functions, operators become a natural extension to defined functions.

TERMINOLOGY: FUNCTIONS VERSUS OPERATORS

Over the years, there has sometimes been confusion between the terms "function" and "operator". The terms have sometimes been used interchangeably. In APL, it's useful to differentiate the terms.

A *function* is that which takes in one or more *data* objects (or "arguments") and returns new *data* (results). An example of a monadic (single-argument) function is `'1'-'1 3'` takes in one piece of data (the argument `'3'`), and returns new data in the form of the result, `'1 2 3'`. An example of a dyadic (two-argument) function is `'+'-'2+3'` takes in two arguments and returns new data in the form of a result, `'5'`.

An *operator* is that which takes in one or more data objects *or functions* and returns a new *function*. An example of a symbol that can be used as an operator is `'/'-'+'/'` takes in a function (plus) and returns a new *derived* function; in this case, summation. That new derived function, then, acts like any other function. It takes in new data (e.g., `'+/1 2 3'`) and returns new data (`'6'`).

It can be said, then, that the role of functions is to manipulate data, and the role of operators is to manipulate functions.

Both functions and operators may be user-defined in IBM's APL2 implementation. This paper will discuss the recently-introduced defined operators.

OPERATOR DEFINITION

As with a defined function, there are three ways in which a defined operator can be established in an APL workspace:

- 1) It can be loaded or copied from a stored workspace using a system command.
- 2) It can be established in execution mode, using the system function “Fix” ($\square FX$), either by a direct keyboard entry or by another defined function.
- 3) It can be established in function definition mode, using one of the system’s “del”-editors.

Regardless of which facility has been used for establishing an operator, its definition can be displayed or modified in either the function definition mode, in which certain editing capabilities are built-in, or by the combined use of the system functions “Canonical Representation” ($\square CR$) and “Fix”($\square FX$).

Function and operator headers

The *valence of a function* is defined as the number of explicit arguments which it takes. A defined *function* may have any one of six forms of header, as follows:

Function Header: $FN \leftrightarrow$ the function name $L \leftrightarrow$ the left argument $R \leftrightarrow$ the right argument $Z \leftrightarrow$ the explicit result			
Type	Valence	With Result	No Result
Niladic	0	$Z \leftarrow FN$	FN
Monadic	1	$Z \leftarrow FN \ R$	$FN \ R$
Dyadic	2	$Z \leftarrow L \ FN \ R$	$L \ FN \ R$

All dyadic defined functions in APL2 are ambi-valent; that is, they may be called either with or without the left argument (the same as the functions on SHARP APL).

The *valence of an operator* is defined as the number of explicit operands which it takes. Its derived function, however, may have a different valence. A defined *operator* may have any one of eight forms of header, as follows:

Operator Header: $OP \leftrightarrow$ the operator name $L \leftrightarrow$ the left array argument $R \leftrightarrow$ the right array argument $F \leftrightarrow$ the left function operand $G \leftrightarrow$ the right function or array operand $Z \leftrightarrow$ the explicit result			
Number of Arguments/Operands		With Result	No Result
1	1	$Z \leftarrow (F \ OP) \ R$	$(F \ OP) \ R$
1	2	$Z \leftarrow (F \ OP \ G) \ R$	$(F \ OP \ G) \ R$
2	1	$Z \leftarrow L \ (F \ OP) \ R$	$L \ (F \ OP) \ R$
2	2	$Z \leftarrow L \ (F \ OP \ G) \ R$	$L \ (F \ OP \ G) \ R$

Operators are never ambi-valent, although their derived functions may be.

ARGUMENTS AND OPERANDS

The names in the header of a defined function which pass data into that function are called its *arguments*. The names in the header of a defined operator which pass functions or data into that operator (entered within parentheses in the header) are called its *operands*.

These parentheses are used to delimit the *operator* and its value from the *function* and its value. The parentheses are optional at call time. They may be used for clarity when the operator is called; they are normally not required, although including them is never wrong.

PRECEDENCE

This can be summarized with these three simple rules:

1. There is no precedence among functions.
2. There is no precedence among operators.
3. All other things being equal, operators have precedence over functions.

A brief review of functions

Before we try to define operators, however, let's review what functions are. This discussion will be limited to monadic and dyadic functions which return explicit results. They are, after all, often the most useful ones, because one function's result can be directly used as the argument to another functions.

A monadic function which returns an explicit result typically modifies its array argument. The result might be completely different from the argument, or it might be very similar.

```

      ▽ Z←NEGATIVE N
[1]   Z←-N
      ▽

```

```

      NEGATIVE 20
-20

```

Similarly, a dyadic function which returns an explicit result typically combines its two

array arguments in some way to make a new array. The result might be completely different from, or very similar to, either or both of its arguments.

```

      ∇ Z←A PLUS B
[1]  Z←A+B
      ∇

```

```

      10 PLUS 20

```

```

30

```

WHAT'S AN OPERATOR?

An operator is to a function what a function is to an array. Operators can be used to study and manipulate functions, just as functions can be used to study and manipulate arrays. Functions are sometimes likened to verbs, as arrays are sometimes likened to nouns. If we were to continue that analogy, operators would become the adverbs.

One difference between the definition of an operator and a regular APL function is that one or two of the parameters in the operator's header may be functions instead of variables (which, after all, are only named arrays). The *derived functions* defined by the operator can thus control the execution of its function operands on its array arguments.

Another difference between a defined operator and a defined function is the way they are invoked by an APL2 expressions. Operators are only invoked when they are found in the syntactical context of an operator—we'll look at some examples of this context in just a moment.

Suppose that we have an operator called *REDUCTION*, which, for purposes of illustration, will do the same thing as the primitive "reduce" operator:

```

      ∇ RESULT←(FUNCTION REDUCTION) ARGUMENT
[1]  RESULT←FUNCTION/ARGUMENT
      ∇

```

```

      +REDUCTION 10 20 30

```

```

60

```

And, sure, you can use a defined function with a defined operator—here's the *PLUS* function from the previous page:

```

      PLUS REDUCTION 10 20 30

```

```

60

```

This *REDUCTION* operator is monadic, because its only operand (between the parentheses) is *FUNCTION*. You can think of *REDUCTION* as the name of the operator, and *(FUNCTION REDUCTION)* as the name of the derived function that it represents. There is an explicit *RESULT*, and the derived function is monadic because there is only one *ARGUMENT*.

An operator always has *one* set of parentheses in its header to indicate the operator context in which it is to be recognized. A monadic operator takes a single function operand on the *left* of the operator name. A dyadic operator takes a function operand

on the left, and either a function or an array operand on the right of the operator name. Outside the parentheses, the header of a defined operator is exactly the same as for a monadic or dyadic function. It must have a right argument, and it *may* have a left argument, an explicit result, and local variables.

RESULTS ← 0.1 + *RECIPROCAL VECTOR*

What you type to call it

What you typed to define it

```

▽ Z ← L (F RECIPROCAL) R
[1] Z ← ÷ (÷ L) F ÷ R
▽

```

An operator takes as its left operand the function to its left, which may itself be a derived function produced by another operator.

Let's poke around a bit inside of this next operator, and see what that function looks like:

```

▽ Z ← (F SHOW) R
[1] ' ⍎NC F: ', ⍎NC 'F'
[2] ' ⍎CR F: ', ⍎CF 'F'
[3] ' ρ⍎CR F: ', ⍎CR 'F'
[4] ''
[5] Z ← F R
▽

```

Let's just display some information

← ...before we apply the function

[' '⍎' ' is APL2's "enclose" function, turning its argument into a scalar.]

```

×REDUCTION SHOW 10 20 30
⍎NC F: 3      "F" is indeed a function
⍎CR F:        but we can't display it...
ρ⍎CR F: 0 0    ⍎CR returns an empty display

```

6000

In this case, F is a *derived function* (“ \times REDUCTION”). We can only take the canonical representation of a defined function, not a primitive function or a derived function. So, let’s try it with a defined function—we can use that *NEGATIVE* function that we showed a couple of pages back:

```

NEGATIVE SHOW 10 20 30
⊠NC F: 3
⊠CR F:  Z←NEGATIVE N
        Z←-N
ρ⊠CR F:  2 9
-10 -20 -30

```

In fact, we can go a step further:

<pre> SΔSHOW←1 NEGATIVE SHOW 10 20 30 SHOW[1] ∇F[⊠]∇ ∇ Z←NEGATIVE N [1] Z←-N ∇ </pre>	<p>Set “Stop Control”</p> <p>If the function is a defined function, it can be displayed by using $\boxplus CR$, $\boxplus TF$, or one of the system’s del-editors.</p>
---	--

Using an operator to examine a function

Just as a defined function can be written to examine an array, a defined operator can be written to examine the behavior of a function. For example:

```

∇ Z←L (F TRACE) R
[1] 'RIGHT ARGUMENT: ',cR
[2] ' LEFT ARGUMENT: ',cL
[3] Z←L F R
[4] '          RESULT: ',cZ
[5] ''
∇

```

```

(2×3) +TRACE 4×5
RIGHT ARGUMENT:  20
LEFT ARGUMENT:   6
          RESULT: 26

```

Again, that “c”—symbol is APL2’s “enclose” function, transforming its argument into a scalar.

Using an operator to examine another operator

We could even use an operator to tell us something about how another operator works!

```

+TRACE/ 1 2 3 4
RIGHT ARGUMENT: 4
LEFT ARGUMENT: 3
RESULT: 7

RIGHT ARGUMENT: 7
LEFT ARGUMENT: 2
RESULT: 9

RIGHT ARGUMENT: 9
LEFT ARGUMENT: 1
RESULT: 10

```

10

Here's an example of using a defined operator that takes advantage of APL2's default form of display to let us *see* how the function works:

```

▽ Z←L (F SEE) R
[1] ⍝DISPLAYS ARGUMENTS TO AND OPERATION OF
[2] ⍝ ANY FUNCTION SPECIFIED
[3] →(0=⊂NC 'L')/MONADIC
[4] DYADIC:Z←L F R
[5] ⊂←(⊂Z), '↔', (⊂L), 'f', ⊂R
[6] ⊂←''
[7] →0
[8] MONADIC:Z←F R
[9] ⊂←(⊂Z), '↔ f', ⊂R
[10] ⊂←''
▽

```

```

A←(3 4ρ12)+SEE 3 4ρ12?12
12 5 9 9 ↔ 1 2 3 4 f 11 3 6 5
12 14 9 18 5 6 7 8 7 8 2 10
10 19 15 24 9 10 11 12 1 9 4 12

```

```

B←1 2 3 °,×SEE 4 5 6
4 5 6 ↔ 1 2 3 f 4 5 6
8 10 12
12 15 18

```

```

C←2+SEE 2 2+SEE 2
4 4 ↔ 2 2 f 2

6 6 ↔ 2 f 4 4

```

Using an operator to modify a function

More often, a defined operator can be used to *modify* the behavior of a function in some systematic way. For example, we could use a dyadic operator to control the *index origin* during the execution of an arbitrary function—here is such an operator, called “*ORIGIN*”:

```

      ∇ Z←L (FUNCTION ORIGIN IO) R;∇IO
[1]   ∇IO←IO
[2]   Z←L FUNCTION R
      ∇

      10 20 30 1 40 20 10
4 2 1

      10 20 30 (1ORIGIN 0) 40 20 10
3 1 0
```

Notice that the right operand of this operator was a zero in this example—with any operator, the left operand may be any primitive, defined, or derived function—the right operand may be that *or data*.

Notice also that in this example, we used parentheses around the derived function. Without parentheses, the “0” would have been considered part of the three-element vector “40 20 10”, *all four* elements of which would have been taken as the right *operand* of “*ORIGIN*”, with nothing left for the right *argument*. A derived function with no right argument would have been a *SYNTAX ERROR*. If the right operand of a dyadic operator is an *array*, then *something* must always separate the right operand from the right argument; that “something” is the parentheses. If you write the calling expression so that it looks like the header of the operator, with the parentheses in the same places, it will always work. They are often not needed in the calling expression—like each of the preceding examples—but they can always be used.

Hexadecimal operations

Sometimes it’s helpful to be able to work directly with hexadecimal calculations. APL does such an admirable job of converting everything to decimal that it’s often difficult to get the system to handle your hexadecimal calculations at all! However, given the two standard hex-conversion functions, things can be made to work:

```

      ∇ Z←DTH D;∇IO
[1]   ∇DECIMAL TO HEXADECEMAL
[2]   ∇IO←0
[3]   Z←⊘(8ρ16)⊔D
[4]   Z←'0123456789ABCDEF'[Z]
      ∇

      ∇ Z←HTD H;∇IO
[1]   ∇HEXADECEMAL TO DECIMAL
[2]   ∇IO←0
[3]   Z←'0123456789ABCDEF'⊔H
[4]   Z←16⊔⊘Z
      ∇
```

However, entering all of those cumbersome expressions like $DTH (HTD '1A') + HTD '1B'$ just to do a simple addition will eventually take its toll. Sure, you can define an “*ADDHEX*” function, but how about letting you use all of the rest of the functions? You don’t have to write a unique function for every fresh piece of data that you use—why should you have to write unique hex-conversion for every function that you wish to use?—An operator can help:

```

      ∇ Z←L (F HEX) R
[1]  ∇ APPLY FN 'F' TO HEX VALUES IN 'L' AND 'R'
[2]  Z←DTH (HTD L) F HTD R
      ∇
      '1A' +HEX '1B'
00000035

      '1A' ×HEX '1B'
000002BE

```

Supplying two functions


Dyadic operators taking functions as both operands often combine them in some way to produce a new derived function. The primitive inner product operator is a good example. By defining a dyadic operator, we have the opportunity to create similar types of constructions in endless variety.

Let’s write our own version of the “dot”-operator:

```

      ∇ Z←L (F DOT G) R
[1]  Z←L F.G ⍉R
      ∇

```


 We’ll give ours a built-in transpose. And, yes, it’s monadic transpose, because “*G*” is a function.

```

      M
1 2 3 4
5 6 7 8
      M∧.=⍉M
1 0
0 1
      M ^DOT= M
1 0
0 1
      M+DOT×M
30 70
70 174

```

We could also write a defined operator which, instead of combining two functions, just joins their results and returns the two results as a two-element vector, so that you can visually compare them.

```

      ∇ Z←(F AND G) R
[1]  Z←(←F R), (←G R)
      ∇

```

Notice that “*AND*” is a dyadic operator, because it has both left and right operands, “*F*” and “*G*”, but its derived function is monadic, because it takes only a right argument, “*R*”.

```

      ∇ AND ∇ 1 2 3 4 5
1 2 4 3 5 5 3 4 2 1

```

We can use it to compare two derived functions!

```

      ≠\ AND (<\) 0 0 1 1 1 0
0 0 1 0 1 1 0 0 1 0 0 0

```

I.P. Sharp primitive operators

I.P. Sharp announced several new composition operators a while back. Let's see how we would proceed to define those ourselves in APL2. SATN-41 (dated 20 June 81) describes the dyadic case of their "ON" operator as this:

$$A \overset{\cdot\cdot}{F} G B \leftrightarrow (G A) F (G B)$$

In APL2, we can define our own version of that by simply keying in just what their example shows (with the addition of another operator to approximate their item-wise operations for close composition):

```

      ∇ Z←A (F ON G) B
[1]  Z←(G A) F⋄ (G B)
      ∇

```

The "⋄"-operator is "Each", as described in T. More's papers (termed "item-wise" by Ghandour and Mezei).

```

      4 ρ + 1 2 3
1 2 3 1

```

```

      4 ρON+ 1 2 3
1 1 1 1 2 2 2 2 3 3 3 3

```

```

      X←<1 2 3
      X ×ON> X
1 4 9

```

In similar fashion, SATN-41 also shows the "OVER" operator as:

$$A \overset{\cdot\cdot}{F} \overset{\cdot\cdot}{G} B \leftrightarrow F A G B$$

We can define our OVER operator like this:

```

      ∇ Z←A (F OVER G) B
[1]  Z←F⋄ A G B
      ∇

```

```

      N←2 4ρϕ,M←2 4ρ1 2 3 4 5 6 7 8

```

```

      M |OVER- N
7 5 3 1
1 3 5 7

```

```

      R←0 <OVER+ M

```

```

      ρR
2 4

```

```

      R
1 2 3 4
5 6 7 8

```

Definition of a “*WITH*” operator could proceed in the same way, given the existence of an “inverse” operator.

These approximations of the I.P. Sharp composition operators are meant to be only approximations for illustrative purposes. That’s because there are differences between the two systems with respect to *enclose* and *disclose* and the treatment of nested arrays. And, yes, I realize that the SATN states that these composition operators place the axes of the individual results last. But what we’re discussing here is the operator facility, not *enclose*. For simple cases, these defined operators do work, and they provide the means for experimentally altering the behavior of both the functions and the operator facility itself.

Doing the housekeeping

Finally, just as with defined functions, we need to keep track of what we have, so—just as with functions—we have a command for displaying the names of the operators:

```
      )FNS
DTH      HTD      NEGATIVE      PLUS

      )OPS
AND      DOT      HEX      ON      ORIGIN      OVER
RECIPROCAL      REDUCTION      SEE      SHOW
TRACE
```

To summarize the differences between functions and operators:

1. A monadic operator will have its single operand on the left.
2. One or both operands of an operator will be a function.
3. The result from an operator will be a function.
4. Operators have a higher precedence than functions.

These have been just a few scattered examples of how defined operators can be used. Their power stems from the fact that just a small number of operators and functions can be combined to produce *many* derived functions. Their uses are as unlimited as the uses of defined functions.

ACKNOWLEDGEMENTS

This paper is based upon original material by Dave Rabenhorst of IBM’s T.J. Watson Research Center in Yorktown Heights, New York, and upon an in-house IBM presentation by Ray Polivka of IBM’s Mid-Hudson Valley Education Center in Poughkeepsie, New York.

This material has also been recently published in the IBM manual, *An Introduction to APL2* (IBM Corporation, 1982, Order Number SB21-3039).

References

1. Brown, J.A. *A Generalization of APL*. Doctoral Thesis, 1971. Department of Computing and Information Science, Syracuse University. New York: Clearing House 74h004942 AD-770488.
2. Brown, J.A. "APL Language Extensions." *Proceedings of SEAS 1978 Anniversary Meeting*. Stresa, Italy: Volume 1, pp. 335-353.
3. Brown, J.A. "Evaluating Extensions of APL." *APL Quote-Quad*. Vol. 9, No. 4 - Part 1, June 1979, pp. 148-155.
4. Falkoff, A.D., and K.E. Iverson. *APL Language*. Form No. GC26-3847, IBM Corporation, 1975.
5. Falkoff, A.D., and K.E. Iverson. *APLSV Version 3 User's Guide*. Form No. SH20-9087, IBM Corporation, 1976.
6. Ghandour, Z. and J. Mezei. "General Arrays, Operators, and Functions." *IBM Journal of Research and Development*. Volume 17, No. 4, July 1973.
7. Iverson, K.E. *A Programming Language*. New York: John Wiley and Sons, 1962.
8. Iverson, K.E. "Operators and Functions." *IBM Research Report #RC-7091*. T.J. Watson Research Center, Yorktown Heights, New York, 26 April 1978.
9. Iverson, K.E. "Operators." *ACM Transactions on Programming Languages and Systems*. October 1979.
10. Iverson, K.E. "Operators and Enclosed Arrays." *Proceedings of The 1980 APL Users Meeting*. I.P. Sharp Associates, October 1980.
11. Iverson, K.E. "Composition and Enclosure." *I.P. Sharp Associates SATN-41*, 20 June 1981.
12. McGrew, J. *An Introduction to APL2*. Form No. SB-3039, IBM Corporation, 1982.
13. More, T. "Axioms and Theorems for a Theory of Arrays." *IBM Journal of Research and Development*. Volume 17, No. 2, 1973.
14. Rabenhorst, D. *The APL2 Language Manual*. Form No. SB21-3015, IBM Corporation, 1982.

Additional Reading on APL2

An Introduction to APL2 (SB21-3039)

Audience: All users of APL2

This manual is intended to provide an informal introduction to the APL2 system. It contains information, useful to both novice and experienced APL users, on the mechanics of using the system, on writing effective programs, and on a wide range of applications of APL2.

The APL2 Language Manual (SB21-3015)

Audience: All users of APL2

This manual provides a formal description of all of the features and facilities of APL2, providing reference for the details of usage for all of the primitive functions and operators.

The APL2 for CMS Terminal User's Guide (SB21-3014)

Audience: Users of APL2 on CMS systems

This manual provides procedural and reference information about APL2 when it's operated under the CMS environment. It contains detailed information about the terminals that can be used with APL2 under CMS, and the procedures that must be followed in starting a terminal session. This manual also describes the auxiliary processors available with APL2 and workspaces that are distributed with the product. General information is also provided on workspace conversion.

No previous experience with CMS is required for use of this manual. Familiarity with the APL language, however, is assumed.

The APL2 Program Description and Operations Manual (SB21-2990)

Audience: System Programmers and others who maintain APL2

This manual provides information about the method of operation, program organization, data areas, and control block formats of APL2. It also provides information helpful in reading program listings, and information regarding determining and reporting problems.

Users of this manual should be familiar with the host operating environment.

FINANCIAL CONSOLIDATIONS WITH TIMESHARING

K. K. Brunzman
Assistant Corporate Controller
and
Ed E. Irsch
Financial Systems Accountant
Stewart-Warner Corporation
Chicago, Illinois

Abstract

Consideration is given to the informational requirements of published reporting by reviewing the purpose and content of financial statements. Consolidation policy formulates the concept of report preparation and leads to an approach to consolidation reports. The concept and approach represent the basis for considering timesharing facilities in the consolidation reporting process. If there is a difference between the concept of consolidation reporting and the methodology available in a computer application, there is an inherent limitation on computer usage.

Purpose of consolidated financial statements

The purpose of consolidated financial statements is to present, primarily for the benefit of corporate management, directors, and the shareholders of the Corporation, the results of operations and the financial position of the the parent company and its subsidiaries, branches and divisions (entities), essentially as if the group were a single business enterprise. Such reporting also provides information that is useful to creditors, investors and other users in making rational credit, investment and similar decisions, and for regulatory requirements. There is a presumption that consolidated statements are more meaningful than separate statements and that they are usually necessary for a fair presentation when one of the companies in the group directly or indirectly has a controlling financial interest in the other companies.

Content of consolidated financial statements

Entity financial reports included in the consolidated financial statements are structured within a uniform chart of accounts, based on general accounting instructions, accounting policies and procedures, standardized reporting package, designated accounting periods and scheduled reporting dates. The monthly financial package comprising entity statements is submitted uniformly to the Corporate Controller each month.

The consolidated financial statements include the accounts of the Corporation and all of its majority-owned subsidiaries. The Corporation also owns between 20% and 50% of the outstanding shares of distributorships, which are accounted for under the equity method.

The usual condition for a controlling financial interest is ownership of a majority voting interest, and therefore, as a general rule, ownership by one company, directly or indirectly, of over fifty percent of the outstanding voting shares of another company is a qualification for consolidation.

Consolidated financial information is prepared and presented in a manner that is meaningful and most useful and suitable to the needs of the establishment, but not burdened with unnecessary detail. Even though the group of companies is heterogeneous in character (manufacturing, distributive, domestic, foreign, variety of products, etc.) a full consolidation is presented at regular intervals, i.e., monthly, quarterly and annually. In addition, separate statements (subconsolidations) are prepared on demand for various management, legal, or regulatory requirements, for a selected group of entities: domestic, foreign, or both.

Consolidated statements for any specific period may include financial data for the parent and any separate entity, in some instances with a difference in fiscal periods. Such data for the separate entity must, however, represent a fiscal period that closely approaches the fiscal period of the parent—usually a difference of not more than three months—provided intervening events do not materially affect the financial position or results of operations.

Consolidation policy and records

Eliminations (Interentity): In the preparation of consolidated statements, intercompany balances and transactions are eliminated, including open accounts, sales, interest, dividends, etc. As consolidated statements are based on the assumption that they represent the financial position and operating results of a single business enterprise, such statements do not include gain or loss on transactions among the companies (interentity) in the group. Accordingly, intercompany profit or loss on assets remaining within the group is eliminated; the concept applied for this purpose is gross profit or loss.

In the preparation of consolidated statements, the parent company's investment accounts and the subsidiaries', divisions' and branches' equity accounts are offset and eliminated. Differences between the cost of investments on the parent's books and net assets of acquired companies are adjusted against appropriate investment and liability or retained earnings accounts so that total investment values are eliminated against net assets values. An acquired company's earned surplus or deficit at date of acquisition is not included in consolidated earned surplus. Unless materially affecting the consolidated results of operations, an acquired company's results of operations are included as though it had been acquired at the beginning of the year.

Foreign currency statements, transactions and historical adjustments: All assets, liabilities, revenues and expenses that are measured or denominated in foreign currency are translated into equivalent U.S. dollars in order to incorporate foreign currency statements and transactions of foreign entities into consolidated financial statements. An exchange rate is established at each year end, which closely approximates the year and published rate, for the purpose of converting all final balance sheet accounts to

equivalent U.S. dollars. Revenues and expenses are converted throughout the year at average exchange rates.

Preparation of consolidated financial reports

The major difference between consolidated financial reports and consolidated financial statements is the greater amount of detail furnished in reports.

Consolidated financial statements present results of the single business enterprise; consolidated financial reports present results as a single business enterprise and supporting business segments—line of business, group by location, separate tax entities and other definitions.

The preparation of consolidated reports (as a single enterprise or as business segments) includes some form of listing or consolidating of the supporting detail of each accounting entity and the proper selection of interentity accounts and transactions to be eliminated.

Consolidating reports by item are commonly requested on a comparative basis—sales, operating profits, inventories and other selected analyses. Consolidating financial reports and comparative consolidated reports are also commonly requested.

Approaches to preparation of consolidating reports

In a strictly manual system, it is obvious that an amount may be used—and necessarily copied—in several reports. The use of a computer decreases the chance of error in recopying and recalculating.

In a strictly manual system, all data is viewed as data of one enterprise—associating an amount with an item and entity is a convenience of identification to enhance flexibility of reporting. If computer identification of amounts restricts flexibility of reporting, a compromise is reached: use a computer for reporting consolidation information but not for auxiliary reports if a manual approach is more cost effective.

The computer system used in preparing selected reports is a consolidation application in a timesharing package program environment. The user has little knowledge of APL (somewhere around the dangerous level) so writing its own specific programs for the application is not considered. Additions to the package program will not be developed unless there is a somewhat universal application. Any program will not be used if a manual approach is more cost effective. The situation is described not as a stand-off attitude of the parties but as an economic reality. It is (or should be) the same reality as if both parties were departments of the same firm.

The more detailed information which follows is a search for a way of expressing one firm's auxiliary consolidation reporting needs so that a computer can be cost effective.

Consolidating reporting process

Basic consolidation reports: The basic consolidation reports may be seen in any annual report: Statement of Financial Position with supplementary information in the

Statement of (Changes in) Retained Earnings, Statement of Changes in Financial Position and Income Statement.

The Income Statement is generally considered as the status for a period of time of the accounts reported; however, it is also an explosion of one line of the Statement of (Changes in) Retained Earnings and is reported in the Statement of Changes in Financial Position. Therefore, it is associated with reporting changes.

The Statement of Financial Position is generally considered as the status on a specified date of the accounts reported; it could also be described as the net of accumulated changes.

The user has the understanding that it is efficient for APL to consolidate data from independent sources (nodes) when it is consistent with the desired report (form).

Conclusion: Require each independent source (node) within the consolidation to submit data in the format of the reports rather than build reports by selecting changes to be aggregated.

Basic comparisons: The basic comparative report is a selection of items from a similar time frame. Balance Sheets are compared at similar dates or the beginning and end of periods under study. The "Changes in ..." reports are compared by periods of time (by month within the year, by quarters within the year, by year to date, or by year to year).

Retention of accounting data is by period of time. After a stipulated period of time, month to month comparisons to prior years of all changes and balances may be unnecessary; after a longer stipulated period of time, quarter to quarter comparisons of other changes and balances may lose their value; the annual data of some changes and balances may be retained longer than ten years.

Whether or not individual changes and balances are removed from a computer system, the retention of accounting data for reporting is oriented to time segments.

The user has the understanding that it is easier for APL to manipulate data within a framework of time where forms are consistent but the number of nodes is open-ended for additions and deletions rather than consider a time segment as a node with an inconsistent number of consolidating units.

Conclusion: Accept the restriction (the APL program shouts "discipline") that nodes of data will be by consolidating unit and not by date/time segment.

Auxiliary consolidating reports: Auxiliary reports may be described as displays of consolidating data or the support for a consolidated amount. The report may be required for a consolidated amount of change or to support the balance position. Auxiliary reports, when manually prepared, have customized descriptive headings relating to the item(s) and timeframe(s) in the report.

When the consolidating report is for one period of time and contains many items, the items are lines and the consolidating amounts are in columns. When the consolidating report compares the status of an account at various periods of time, consolidating amounts are lines and the time segments are columns. Differences between time segments are required.

Figure 1

STEWART-WARNER CORPORATION
REPORT BY ITEM (IN THOUSANDS OF DOLLARS)
ITEM: INVENTORIES

ENTITY (NODE)	3/82	2/82	13/81	3/81	3/82	3/82	3/82
					MINUS	MINUS	MINUS
					2/82	13/81	3/81
2	42,309	41,967	42,652	37,073	342	-343	5,236
5	22,049	22,553	22,715	21,029	-504	-666	1,020
7	11,721	11,732	12,170	11,111	-11	-449	610
24	1,240	1,163	1,408	1,460	77	-168	-220
8	3,105	3,098	3,065	2,825	7	40	280
9	12,432	12,650	13,373	15,764	-218	-941	-3,332
22				79			-79
23							
11	4,895	4,917	5,125	5,004	-22	-230	-109
6	2,303	2,272	2,172	1,928	31	131	375
14	7,252	7,263	7,333	7,498	-11	-81	-246
16	3,082	3,100	3,277	5,097	-18	-195	-2,015
17	201	192	255	265	9	-54	-64
18	830	869	886	880	-39	-56	-50
19	188	311	353	319	-123	-165	-131
20	155	162	156	205	-7	-1	-50
21	471	475	565	549	-4	-94	-78
25	12,471	13,017	12,766	14,124	-546	-295	-1,653
3	755	675	675	1,225	80	80	-470
10	14	24	24	60	-10	-10	-46
26							
15							
13							
1	588	588	588	1,066			-478
101	126,062	127,029	129,557	127,561	-967	-3,495	-1,499
102							
103	-7,858	-8,141	-7,976	-8,208	283	118	350
104							
100	118,205	118,888	121,581	119,353	-683	-3,376	-1,148
400	4,616	4,799	5,181	7,353	-183	-565	-2,737

Either type of consolidating report may be requested for the amounts shown in the consolidated annual report or in various independent subconsolidations.

A specialized consolidating report computes turnovers—an accumulated addition to an account divided by the average balance of the account for the same period—and displays the factors and turnovers in the report.

Detailed analysis of a reporting unit's data is required only if the reporting unit cannot

furnish supplementary comparative reports of its own activity; the basic and auxiliary reporting of a consolidation reporting system tend to be concentrated in reporting consolidating support and comparative analysis of a consolidated amount or factor.

The user has the understanding that each desired report of a comparison of an account requires an access to each node of data shown in the consolidating report with an attendant cost related to each access.

Conclusion: Determine justification of computer use for each report or show that the consistency of the reporting requirements might lend itself to selecting various items for the various reports with one access.

Specialized reports: One type of report displays the consolidated amounts of the Statement of Financial Position in a sequence different from the shareholders' report.

Another type of report requires a display of consolidated accounts and percentages to a base number in adjacent columns: a comparative income statement shows selected items in lines and the amount and % to sales as a pair of columns for selected time segments (current month this year, this month last year, year to date this year, and year to date last year).

The user has the understanding that APL reports are generally copied out of forms. If the same base information is reported in two different combinations, two forms are needed. All forms extend over all consolidating units in the system.

Conclusion: If specialized reports are required from the fully consolidated unit only, consider copying the base information into a separate system consisting of the one unit and compute all the reporting forms required.

Multiple subconsolidations: Under ideal circumstances, all basic consolidation reports, basic comparisons and auxiliary consolidating reports may be displayed for any subconsolidation. A subconsolidation is a predetermined set of reporting units; but with multiple subconsolidations, a reporting unit may be in more than one subconsolidation.

Transactions among reporting units of the subconsolidation and the status of specified accounts have an effect on the amounts reported for the subconsolidation. The transactions and status are selected from those affecting the full consolidation reports.

The ability to select these consolidating adjustments and eliminations is considered the heart of subconsolidation reporting; the selection of reporting units is considered conventional.

The user has the understanding that conditional selection and accumulation of items from a list in random sequence is not an efficient APL task.

Conclusion: Consolidating adjustments and eliminations for consolidation and subconsolidation reporting are determined in a manual system.

Processing and reporting: It is a rarity for the user that a report is not required showing the details of a consolidation or subconsolidation of forms. It is a rarity for the user not to consider the input of data (add, check, change, check and calculate subtotals) as one process.

The user has the understanding that there is a cost associated with each accessing of each node for each process or report.

Conclusion: If a node is accessed to compute the consolidated unit and accessed again to display a report of the consolidating and consolidated units, determine justification of consolidations on a monthly basis or show that reports are usually required for any consolidation to encourage combining the consolidation and reporting with one access.

Application of system

The computerized data consists of 21 lines of data calculated from 86 available lines of input and 10 individual combinations aggregated from 24 reporting units. The consolidating data comprises an additional 18 reporting units for 9 consolidations. The ten combinations include the full consolidation as one set and nine combinations organized into three nonoverlapping sets. (see Figure 2)

Eight national currencies are involved requiring translation to the consolidating currency which is U.S. dollars and cents. To provide that all input be made without entering a decimal character, one national currency is described as U.S. pennies.

Three rates are identified each month for each currency: the beginning of the year rate, the average year to date rate, and the end of the period rate. The end of the period rate is applied to all assets and liabilities, the average rate is applied to the income statement and all three rates are used in the equity section of the Balance Sheet.

Customized calculations of rounding line items are performed for assets, liabilities and income. The rounding line items represent the difference between the sum of translated amounts of each line item and the translation of the entered total line item.

Customized grouping of lines are calculated.

Customized calculation of translation adjustments for FASB 52 are performed in the equity section of the Balance Sheet. The beginning net assets of each reporting unit stated in national currency is translated at the beginning of year rate; the year to date transactions (income, dividends, changes in capital) of each entity are translated at average rates, the end of the period net assets is calculated in national currency and translated at end of period rates; the translation adjustment represents the difference between the net translation of the beginning and year to date items and the translation of the end of period item.

The computerized basic consolidation reporting consists of:

- 1) The Balance Sheet, Income Statement and Statement of Retained Earnings for the full consolidation on a monthly basis and
- 2) The Balance Sheet and Income Statement for subconsolidations: one on a monthly basis, one on a quarterly basis and six on an annual basis.

The computerized basic comparisons report consolidates data of selected timeframes (the current month, prior month, and prior fiscal year, and same month last year) and shows three comparisons: current month less prior month, current month less end of prior fiscal year, and current month less same month last year.

Computerized auxiliary reporting is limited to the consolidating data for one account displayed in the columnar format described above; however, selection may be made for any account for the full consolidation.

Future application of system

Some of the possibilities of future applications have been outlined in the basic and auxiliary reporting needs for consolidations and subconsolidations; however, unless all associated computer costs are identified immediately, the user approaches alternative and trial solutions with extreme caution.

Figure 2

Multiple Consolidations Assigned to Non-Overlapping Groups

		SUBCONSOLIDATIONS										
		F	-----									
		U										
		L										
		L	GROUP I					II		III		
			-----					-----		-----		
ENTITIES												
1		X							X	X		
2		X				X			X	X		
3		X				X	X					
5		X							X	X		
6		X							X	X		
7		X			X				X	X		
8		X					X					
9		X		X					X	X		
10		X							X	X		
11		X							X	X		
13		X	X						X		X	
14		X	X						X		X	
15		X	X						X		X	
16		X	X				X			X		
17		X	X				X			X		
18		X	X				X			X		
19		X	X				X			X		
20		X					X			X		
21		X					X			X		
22		X		X					X			
23		X		X					X			
24		X			X		X		X			
25		X				X			X			
26		X							X			
COMBINATIONS												
A		101	201	501	601	801	411	901	301	401	701	
MANUAL SYSTEMS												
ELIM		102	202	502	602	802		902	302	402	702	
ADJ		103	203	503	603	803		903	303	403	703	
(HIST ADJ)		104	204	504	604	804		904	304	404	704	
CONSOLIDATIONS												
A		100	200	500	600	800		900	300	400	700	

USING THE BOX AND JENKINS METHOD FOR THE ANALYSIS OF EUROPEAN AIR TRAFFIC

Alain Wilemme
Route Programming and Corporate Planning Division
Air France
Paris, France

This paper originally was the report on a study carried out at the end of 1981. The Air France Route Programming and Corporate Planning Division had been asked to quantify the effect on traffic of a policy introduced by Air France on most of its European routes in April 1981. This new policy was mainly characterized by lower fares and simplified inflight service. The effect can be judged from the graph in Appendix 1. Air France's traffic clearly showed a jump in April 1981 whereas the traffic of other European carriers continued at the same level. Nevertheless, although present traffic can be compared with the past traffic, just as various carriers can be compared with each other, we do not know what the situation would have been in the absence of the innovating policy. There was, consequently, no reference point authorizing the measurement of gains in traffic for Air France and of possible losses for competitors. Specifically, the comparison of the existing situation with a previous situation affected by the economic crisis was not completely convincing. Because any traffic decrease can be followed by a recovery, comparing the summer of 1981 with the summer of 1980 might have led to an overly optimistic conclusion; on the other hand, referring to the summer of 1979 might have led to unjustified pessimism due to the excellent level of traffic during that period.

The only correct measurement consisted of comparing traffic without innovation; this resulted in an operation consisting of building up simulated air traffic figures without any reference to the innovations applied in the spring of 1981.

1. METHODOLOGY

1.1 Scope of the study

Simulation can be made of all European traffic. However, all traffic patterns were not affected by innovations or by competition from other air carriers. Accurate measurement required the selection of relevant city-pairs. For efficiency's sake, the traffic patterns were not analysed separately but within routes grouping France/European country city-pairs. The method used took quite a long time to implement, and only three countries were finally analysed.

For reasons of confidentiality, the letters A, B, and C will be used to denote these

countries in this paper. Furthermore, Air A, Air B, and Air C will denote the main carrier of each country.

Only traffic patterns on which Air France and its competitors had both been present since January 1, 1975 and before and after April 1, 1981 were considered.

1.2 Data used

The data for each country considered consisted of three monthly chronological series, beginning in January 1975, and ending on the date of the last data known at the time of this study, i.e., October 1981. The series respectively covered:

- Air France
- its main partner airline and all carriers other than Air France and the main partner
- all traffic

For each series the monthly traffic involved consisted of the number of passengers carried by the carrier or carriers considered on all patterns selected for the country under study. The third series represents the sum of the first two, and it would theoretically have been possible to dispense with it. Nonetheless, the three series were the subject of three separate analyses in order to have the means of verifying the coherence of the results obtained. Appendix 2 lists all the series studied.

1.3 Principle of the simulation

The only data used covered traffic and excluded fare and economic data. It must be pointed out, however, that the past evolution of traffic was to a certain extent the reflection of the successive states of the general context—economic activity, price structure, etc. On the assumption that no external event occurred to modify the evolution of traffic after March, the method used proposed to describe what the traffic would have been after March 31 on the basis of the elements constituting this traffic between January 1, 1975 and March 31, 1981. Appendix 3 contains several elements used in this method, known as the Box and Jenkins method, and specifies the notations allowing an experienced user to determine the exact nature of the models used.

Each model provided chronological series representing the probable traffic as of April 1981 in the absence of any innovation. The term “innovation” is to be understood in the sense which the Box and Jenkins model gives to “one step ahead forecast error” [Ref. 1]. If $z_t(1)$ represents the forecast, at date $t + 1$, of the evolution studied in terms of the data known on date t and, if the actual value of the process is fixed at level z_{t+1} , the difference $z_{t+1} - z_t(1)$ represents the one step ahead forecast error of the process at date $t + 1$. The basic objective of this study was to measure the effects of the New European Service; however, other modifications in service also occurred during the same period, including changes in flight times, frequencies and aircraft types. The difference between the traffic observed and the traffic simulated was the consequence of all such modifications and cannot therefore be attributed *a priori* to the New European Service alone, although it is not possible to determine to what degree the

New European Service was in fact responsible. For this reason, the term “innovation”, which is non-committal, is being used in this paper.

2. RESULTS

The results for the three countries studied were presented in a similar manner. A summary will be found in section 4, where the method used to calculate the listed figures is specified. The results themselves will be found in sections 4.1 through 4.3.

2.1 Comparison between observation and simulation

The traffic recorded was expressed as a percentage of the traffic which might have been expected in the absence of any innovation. The monthly results are only approximate, but the averages are, on the whole, rather reliable.

General observations

- air traffic was generated, indeed: the total traffic observed is greater than the total traffic simulated
- the additional traffic benefited Air France
- Air France sometimes won over traffic from its competitors (not in Country B, however; see 3.2)

2.2 Analysis of traffic won by Air France

We listed the 1980 figures for Air France and then so-called normal growth; i.e., the increase in Air France traffic in the **absence of innovation**.

The transfer from the competitors to Air France could be calculated as the difference between the expected traffic of the competitor in the absence of any innovation and the traffic recorded by the same competitor.

The traffic generated (the difference between the total traffic observed, T , and the simulated traffic without any innovation, ST ; that is, TST) was assumed to be won over completely by Air France. The breakdown of the effect of innovation between the traffic generated and the transfer is only meaningful if we suppose that this effect only benefited Air France. This is a reasonable supposition if we think of the New European Service alone; it is perhaps somewhat less reasonable in the overall context.

We must note that the expression “traffic generated” refers only to air transportation; what may have been involved was a transfer from surface transportation to air transportation.

2.3 Other results

The 1981 figures with innovation are the traffic observed; the 1981 figures without innovation are the traffic simulated. The results are given in number of passengers and indices (level summer 1980=100). Air France's share with and without innovation is indicated because one of the main consequences of innovation was that Air France gained a larger share of total traffic. The share of the traffic without any innovation is to be compared with the share of traffic during the previous summer seasons.

3. COMMENTS BY COUNTRY

3.1 Country A (see section 4.1)

Air France's policy was an overall success since, in the course of "normal" events, it should have lost approximately 5,000 passengers, whereas we can estimate at more than 20,000 the number of new passengers, to which must be added almost 14,000 passengers who shifted over from Air A to Air France; Air A thus lost 4% of the traffic it might have expected. However, its traffic was still above that of the previous year.

3.2 Country B (see section 4.2)

Country B is by far the most interesting case studied. Two conclusions must be mentioned:

1. The traffic generated was very high. On this flight beam, and especially on Paris/Country B, trains and planes (not to mention cars) provide rather similar service, and the number of passengers shifting from one means of transportation to another according to modifications (even minor ones) in supply was undoubtedly quite high. The New European Service may therefore have induced many travellers to take the plane.
2. The competition, mainly from Air B, remained at the same level without innovation. The transfer (which was in fact negative) can be considered null. A number of interpretations for this are possible:
 - a) Air B service has no equivalent and cannot be replaced by other existing products; this interpretation seems somewhat simplistic
 - b) Air B limited its capacity in terms of available seats

The potential demand for this carrier may be much higher than the demand revealed by restrictions on the capacity. It is not impossible that part of the potential demand for Air B was diverted to Air France; however, even after innovation, the demand still saturated Air B's capacity. For this reason, Air B's traffic evolved not according to the rate of demand, with all the vicissitudes which this implies, but according to an intrinsic rate defined by its capacity (and therefore quite forecastable by the model).

Without innovation, Air France's traffic would have decreased compared with the same period of the previous year; and its share of total traffic would have been almost at the same level as for the three previous years, whereas Air France gained 4 percentage points.

3.3 Country C (see section 4.3)

A large gain for Air France, which would, nonetheless, have experienced higher growth than that of its competitors even without any innovation (4%). The shift in passengers to the competitors represented one fifth of this gain, but contributed to considerably reducing the situation of the other carriers (Air C and others), as shown by the spectacular increase in the share of Air France traffic.

Appendix 4 contains the mathematical content of the models used.

4 SUMMARY OF THE RESULTS

Traffic from April to October 1981

	Actual		Simulated	
	Monthly	Cumulated	Monthly	Cumulated
Air France	F	$+/F$	SF	$+/SF$
Others	O	$+/O$	SO	$+/SO$
Total	T	$+/T$	ST	$+/ST$

Cumulated traffic from April to October 1980:

Air France. $+/F80$
 Others..... $+/O80$
 Total..... $+/T80$

Comparison between observation and simulation

Observed traffic is compared with the simulated traffic based on the data known before March 1981 **under the assumption** that no innovation occurs after March 1981.

Observed traffic is expressed as a percentage of simulated traffic.

	APR	MAY	JUN	JUL	AUG	SEP	OCT	AVERAGE
Air France	$100 \times (F \div SF)$	$100 \times ((+/F) \div (+/SF))$
Other Carriers	$100 \times (O \div SO)$	$100 \times ((+/O) \div (+/SO))$
Total	$100 \times (T \div ST)$	$100 \times ((+/T) \div (+/ST))$

Analysis of the traffic won by Air France

BREAKDOWN	PASSENGERS	INDEX
Level 1981	$+/F$	left
Traffic generated	$(+/T) - (+/ST)$	column
Transfer from other carriers		expressed
to Air France	$-((+/O) - (+/SO))$	in percentage
"Normal" growth		of level 80
(no new policy)	$(+/SF) - (+/F80)$	
Level 1980	$+/F80$	

Gain for Air France: traffic generated + transfer.

Other results

	Air France		Others		Total	
	pass.	index	pass.	index	pass.	index
Level 1981 with innovation ..	(+/F)	-	(+/O)	-	(+/T)	-
Level 1981 without innovation	(+/SF)	-	(+/SO)	-	(+/ST)	-
Level 1980	+ / F80	-	+ / O80	-	+ / T80	-

Index columns express the passengers column in percentage of the last row.

Average share of Air France's traffic in total traffic

with innovation : $100 \times (+/F) \div (+/T)$

without innovation : $100 \times (+/SF) \div (+/ST)$

4.1 France/Country A

Comparison between observation and simulation

Observed traffic is expressed in percentage of simulated traffic.

	APR	MAY	JUN	JUL	AUG	SEP	OCT	AVERAGE
Air France	112	120	116	114	108	114	110	113.6
Air A	95	98	91	97	96	99	96	95.9
Total	102	108	102	104	101	106	102	103.5

Analysis of the traffic won by Air France

BREAKDOWN	PASSEN- GERS	INDEX
Level 1981	297,162	111.5
Traffic generated	20,746	8.1
Transfer from other carriers to Air France	13,668	5.3
"Normal" growth (no new policy)	-4,763	-1.8
Level 1980	257,511	100.0
Gain for Air France	34,414	13.4

Other results

	Air France pass.	France index	Air A: pass.	A: index	pass.	Total index
Level 1981 with innovation...	287,162	111.5	318,192	101.4	605,354	106.0
Level 1981 without innovation	252,748	98.2	331,860	105.8	584,608	102.3
Level 1980	257,511	100.0	313,778	100.0	571,289	100.0

Average share of Air France's traffic in total traffic

with innovation : 47.4

without innovation : 43.2

4.2 France/Country B

Comparison between observation and simulation

Observed traffic is expressed in percentage of simulated traffic.

	APR	MAY	JUN	JUL	AUG	SEP	OCT	AVERAGE
Air France	125	117	120	124	127	111	108	118.1
Others	97	100	102	101	104	101	98	100.1
Total	108	107	109	110	112	105	102	107.4

Analysis of the traffic won by Air France

BREAKDOWN	PASSEN- GERS	INDEX
Level 1981	260,346	115.8
Traffic generated	40,217	17.9
Transfer from other carriers to Air France	-397	-0.2
"Normal" growth (no new policy)	-4,247	-1.9
Level 1980	224,773	100.0
Gain for Air France	39,820	17.7

Other results

	Air France pass.	France index	Others pass.	Others index	pass.	Total index
Level 1981 with innovation...	260,346	115.8	326,186	105.7	586,532	109.9
Level 1981 without innovation	220,526	98.1	325,789	105.5	546,315	102.4
Level 1980	224,773	100.0	308,696	100.0	533,469	100.0

Average share of Air France's traffic in total traffic

with innovation : 44.4

without innovation : 40.4

4.3 France/Country C

Comparison between observation and simulation

Observed traffic is expressed in percentage of simulated traffic.

	APR	MAY	JUN	JUL	AUG	SEP	OCT	AVERAGE
Air France	130	122	116	107	112	111	108	115.6
Others	75	97	88	105	111	102	106	96.8
Total	104	110	102	106	111	106	107	106.3

Analysis of the traffic won by Air France

BREAKDOWN	PASSEN- GERS	INDEX
Level 1981	412,048	120.0
Traffic generated	44,406	12.9
Transfer from other carriers to Air France	11,189	3.3
“Normal” growth (no new policy)	13,063	3.8
Level 1980	343,390	100.0
Gain for Air France	55,595	16.2

Other results

	Air France		Others		Total	
	pass.	index	pass.	index	pass.	index
Level 1981 with innovation...	412,048	120.0	339,551	98.1	751,599	109.0
Level 1981 without innovation	356,453	103.8	350,740	101.4	707,193	102.6
Level 1980	343,390	100.0	345,978	100.0	689,368	100.0

Average share of Air France's traffic in total traffic

with innovation : 54.8

without innovation : 50.4

These results must be used with some care. But one can conclude that it is almost certain that this summer Air France could have had a traffic equal or superior to the one they actually had if the innovations had followed the same probability distribution as it had in the past.

References

1. Box, G.E.P., and G.M. Jenkins *Time Series Analysis: Forecasting and Central.* Holden Day.
2. Anderson, O.D. *Time Series Analysis and Forecasting: The Box-Jenkins Approach.* Butterworths.
3. Montgomery, D.C., and L.A. Johnson. *Forecasting and Time Series Analysis.* McGraw Hill.

APPENDIX 1

Figure 1

Air France's Traffic in Europe

(A roughly 8 per cent jump can be clearly seen for April 1981)

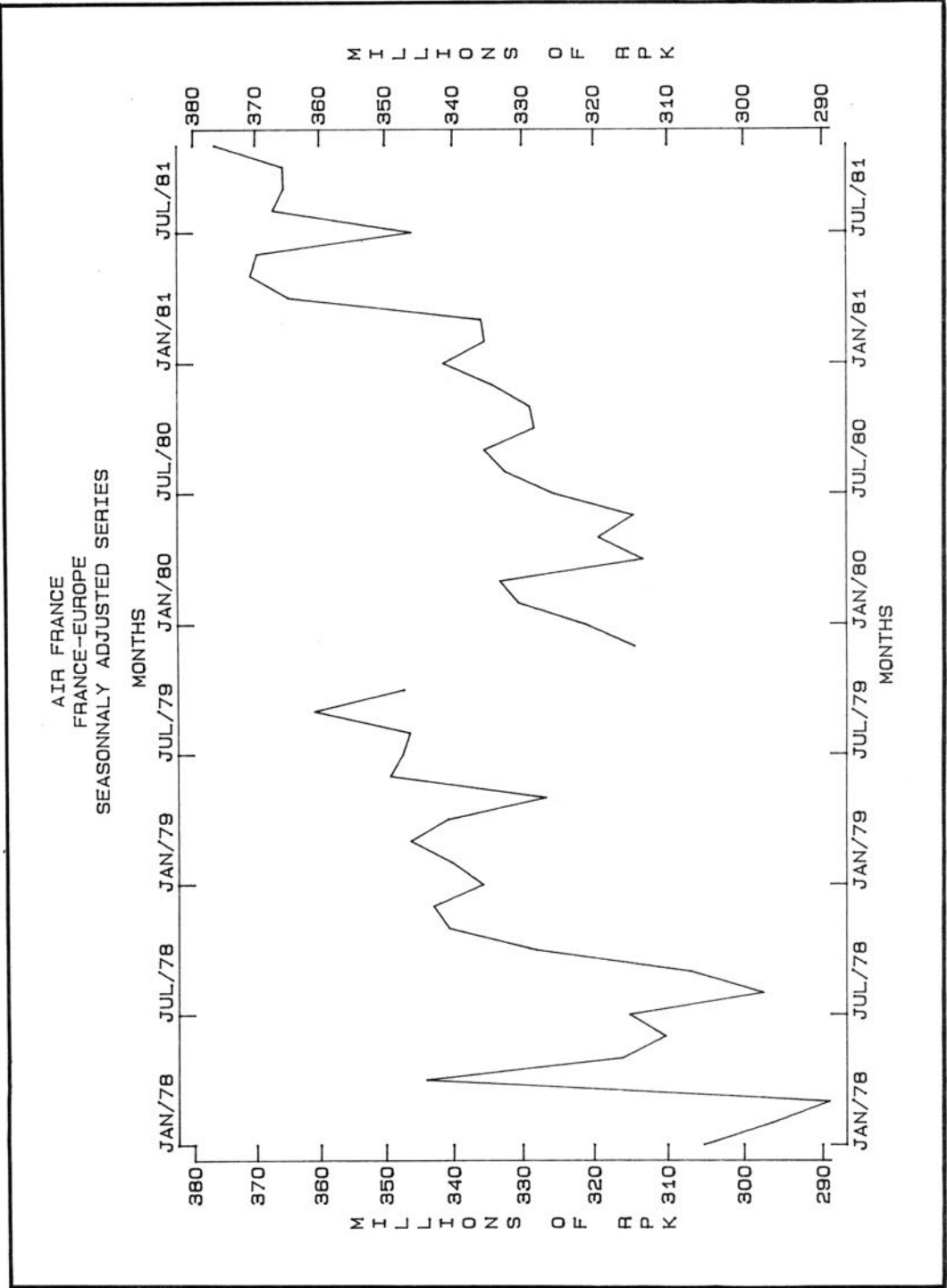
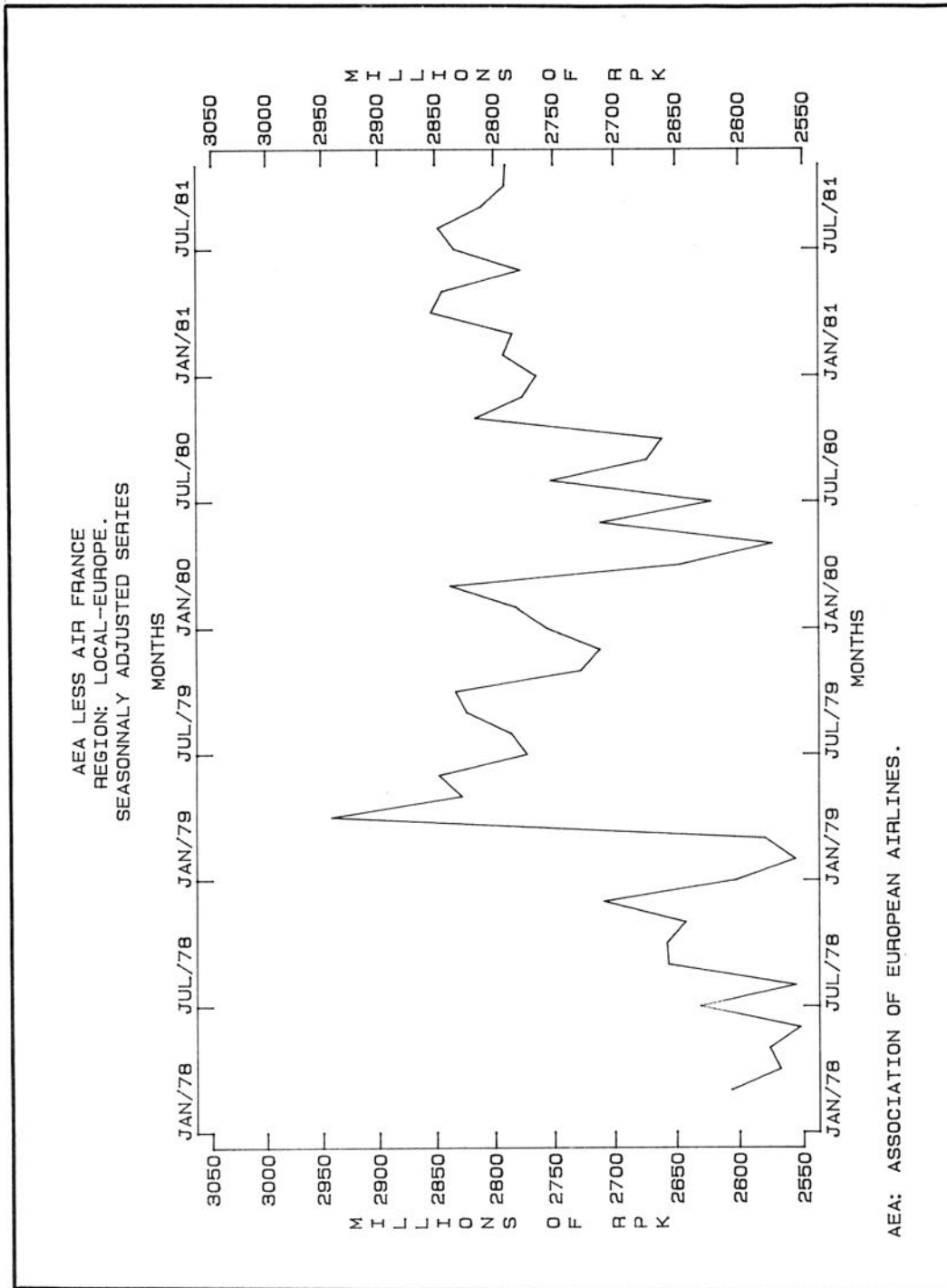


Figure 2

AEA less Air France

(The short dash line represents a jump similar to the one for Air France (with the same growth rate) and shows, by contrast, the efficiency of Air France's policy.)



APPENDIX 2

The series used to build up the models are shown in Figure 3. The share of Air France in total traffic, and the cumulation of the summer figures presented in Figure 4 did not help the selection of the models but allowed us to judge the results of simulations.

Figure 3

Passenger Traffic: Paris/Cities in Country A

AIR FRANCE

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
1975	23,063	22,714	25,366	32,387	29,946	33,592	22,246	18,926	32,477	30,170	22,311	25,372
1976	27,058	29,487	30,818	34,441	36,532	34,917	23,364	18,378	37,457	36,956	35,581	26,811
1977	26,583	29,308	37,740	31,935	37,195	40,575	25,173	18,020	43,346	42,384	37,311	29,755
1978	32,941	31,454	32,503	40,848	38,779	43,043	30,785	20,880	38,881	44,247	42,685	31,415
1979	33,402	34,753	41,322	37,075	43,336	46,403	32,884	28,634	46,877	51,061	29,315	26,446
1980	32,752	34,251	38,739	34,519	37,821	39,433	30,573	25,005	45,466	44,694	38,386	28,159
1981	31,437	33,847	37,226	38,742	45,547	46,920	33,295	25,550	49,173	47,935		

AIR FRANCE + AIR A

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
1975	33,745	34,600	39,202	36,267	35,824	38,726	32,367	30,671	39,076	43,147	40,360	29,961
1976	32,935	36,082	37,577	37,518	42,701	39,317	35,960	35,180	45,117	43,366	40,223	31,062
1977	36,705	35,430	43,967	38,436	44,591	44,148	37,923	36,099	47,983	43,977	32,689	24,351
1978	31,098	34,823	37,848	42,947	43,256	45,179	34,735	30,807	48,475	46,286	43,965	34,479
1979	38,660	38,162	43,254	43,267	49,457	50,038	37,464	34,308	49,698	54,980	33,021	30,347
1980	38,620	40,500	48,871	43,674	46,067	47,234	39,191	34,342	51,583	51,687	45,325	36,387
1981	37,725	39,446	44,022	44,756	48,707	46,524	39,256	35,051	52,035	51,863		

TOTAL

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
1975	56,808	57,314	64,568	68,654	65,770	72,318	54,613	49,597	71,553	73,317	62,671	55,333
1976	59,993	65,569	68,395	71,959	79,233	74,234	59,324	53,558	82,574	80,322	75,804	57,873
1977	63,288	64,738	81,707	70,371	81,786	84,723	63,096	54,119	91,329	86,361	70,000	54,106
1978	64,039	66,277	70,351	83,795	82,035	88,222	65,520	51,687	87,356	90,533	86,650	65,894
1979	72,062	72,915	84,576	80,342	92,793	96,441	70,348	62,942	96,575	106,041	62,336	56,793
1980	71,372	74,751	87,610	78,193	83,888	86,667	69,764	59,347	97,049	96,381	83,711	64,546
1981	69,162	73,293	81,248	83,498	94,254	93,444	72,551	60,601	101,208	99,798		

SHARE OF AIR FRANCE

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
1975	40.6	39.6	39.3	47.2	45.5	46.5	40.7	38.2	45.4	41.2	35.6	45.9
1976	45.1	45.0	45.1	47.9	46.1	47.0	39.4	34.3	45.4	46.0	46.9	46.3
1977	42.0	45.3	46.2	45.4	45.5	47.9	39.9	33.3	47.5	49.1	53.3	55.0
1978	51.4	47.5	46.2	48.7	47.3	48.8	47.0	40.4	44.5	48.9	49.3	47.7
1979	46.4	47.7	48.9	46.1	46.7	48.1	46.7	45.5	48.5	48.2	47.0	46.6
1980	45.9	45.8	44.2	44.1	45.1	45.5	43.8	42.1	46.8	46.4	45.9	43.6
1981	45.5	46.2	45.8	46.4	48.3	50.2	45.9	42.2	48.6	48.0		

Passenger Traffic: Paris/Cities in Country B

Figure 4

AIR FRANCE

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
1975	25,059	26,248	31,580	25,284	26,242	28,718	24,128	24,738	28,610	25,302	22,049	25,246
1976	27,708	29,880	33,252	28,455	29,957	29,643	25,604	22,956	32,654	29,599	28,775	26,725
1977	26,939	30,647	32,533	30,474	32,174	33,637	25,468	21,934	34,576	32,236	26,373	24,456
1978	28,710	28,239	32,713	31,159	31,757	32,992	26,381	18,151	30,683	32,073	30,447	29,770
1979	29,945	32,764	36,523	33,831	34,620	36,695	27,652	22,852	37,332	35,053	18,336	24,105
1980	30,916	32,985	34,533	32,003	31,633	34,387	28,486	24,481	36,227	37,556	27,742	26,678
1981	29,332	32,694	33,254	39,197	37,633	41,294	34,359	29,523	39,502	38,838		

OTHER CARRIERS [INCLUDING AIR B]

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
1975	38,421	38,686	43,762	41,538	39,877	43,508	33,910	29,338	41,657	47,482	36,790	32,596
1976	34,668	39,419	44,095	42,244	42,108	42,575	39,011	34,413	45,846	45,317	38,518	36,643
1977	40,403	41,166	44,993	42,191	45,188	47,174	42,925	37,033	47,733	46,708	43,169	40,992
1978	43,782	40,129	44,720	48,746	46,358	47,530	42,221	35,504	48,210	49,754	39,370	37,208
1979	40,553	42,271	46,250	45,671	47,744	47,956	44,315	38,703	51,615	52,541	31,837	34,097
1980	41,356	42,609	48,238	44,276	43,008	46,083	42,183	37,010	47,401	48,735	43,598	41,980
1981	43,882	44,164	50,424	45,659	46,274	49,298	44,423	39,684	50,622	50,226		

TOTAL

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
1975	63,480	64,934	75,342	66,822	66,119	72,226	58,038	54,076	70,267	72,784	58,839	57,842
1976	62,376	69,299	77,347	70,699	72,065	72,218	64,615	57,369	78,500	74,916	67,293	63,368
1977	67,342	71,813	77,526	72,665	77,362	80,811	68,393	58,967	82,309	78,944	69,542	65,448
1978	72,492	68,368	77,433	79,905	78,115	80,522	68,602	53,655	78,893	81,827	69,817	66,978
1979	70,498	75,035	82,773	79,502	82,364	84,651	71,967	61,555	88,947	87,594	50,173	58,202
1980	72,272	75,594	82,771	76,279	74,641	80,470	70,669	61,491	83,628	86,291	71,340	68,658
1981	73,214	76,858	83,678	84,856	83,907	90,592	78,782	69,207	90,124	89,064		

SHARE OF AIR FRANCE

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
1975	39.5	40.4	41.9	37.8	39.7	39.8	41.6	45.7	40.7	34.8	37.5	43.6
1976	44.4	43.1	43.0	40.2	41.6	41.0	39.6	40.0	41.6	39.5	42.8	42.2
1977	40.0	42.7	42.0	41.9	41.6	41.6	37.2	37.2	42.0	40.8	37.9	37.4
1978	39.6	41.3	42.2	39.0	40.7	41.0	38.5	33.8	38.9	39.2	43.6	44.4
1979	42.5	43.7	44.1	42.6	42.0	43.3	38.4	37.1	42.0	40.0	36.5	41.4
1980	42.8	43.6	41.7	42.0	42.0	42.7	40.3	39.8	43.3	43.5	38.9	38.9
1981	40.1	42.5	39.7	46.2	44.9	45.6	43.6	42.7	43.8	43.6		

Figure 5

Passenger Traffic: Paris/Cities in Country C

AIR FRANCE

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
1975	24,171	26,040	32,356	35,431	39,364	37,436	30,723	34,754	43,610	45,119	31,186	28,788
1976	25,714	28,897	36,503	39,479	44,754	45,303	38,653	26,876	44,942	40,219	36,208	29,964
1977	30,726	33,319	40,277	40,452	41,037	41,193	33,430	28,037	50,312	49,081	37,959	34,743
1978	37,107	36,492	43,148	53,750	48,869	48,677	41,788	29,897	53,704	54,950	46,065	39,860
1979	42,913	47,269	64,509	60,817	59,708	58,241	48,697	37,707	61,793	58,218	41,669	37,568
1980	42,829	44,380	51,034	54,891	54,223	49,357	42,754	31,623	55,426	55,116	46,564	36,772
1981	51,250	44,134	53,166	73,678	69,271	61,084	47,383	36,704	63,383	60,545		

OTHER CARRIERS [INCLUDING AIR C]

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
1975	31,550	33,185	41,176	47,478	38,992	47,984	36,808	31,661	50,798	46,633	37,508	29,694
1976	30,175	34,787	41,403	39,996	40,442	31,458	30,985	35,263	50,617	48,975	34,608	27,739
1977	32,910	31,635	44,627	52,612	56,061	49,141	43,716	37,779	53,910	52,229	41,595	31,753
1978	30,169	28,855	36,084	49,751	52,805	47,566	38,981	33,548	51,416	51,332	39,389	35,375
1979	35,840	32,758	18,625	51,435	57,388	48,266	37,474	32,444	56,641	54,940	32,811	29,013
1980	33,512	36,256	45,169	54,004	57,757	53,228	39,247	34,209	55,030	52,503	42,662	34,953
1981	29,975	36,110	40,734	40,645	56,455	45,737	42,267	38,763	57,752	57,932		

TOTAL

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
1975	55,721	59,225	73,532	82,909	78,356	85,420	67,531	66,415	94,408	91,752	68,694	58,482
1976	55,889	63,684	77,906	79,475	85,196	76,761	69,638	62,139	95,559	89,194	70,816	57,703
1977	63,636	64,954	84,904	93,064	97,098	90,334	77,146	65,816	104,222	101,310	79,554	66,496
1978	67,276	65,347	79,232	103,501	101,674	96,243	80,769	63,445	105,120	106,282	85,454	75,235
1979	78,753	80,027	83,134	112,252	117,096	106,507	86,171	70,151	118,434	113,158	74,480	66,581
1980	76,341	80,636	96,203	108,895	111,980	102,585	82,001	65,832	110,456	107,619	89,226	71,725
1981	81,225	80,244	93,900	114,323	125,726	106,821	89,650	75,467	121,135	118,477		

SHARE OF AIR FRANCE

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
1975	43.4	44.0	44.0	42.7	50.2	43.8	45.5	52.3	46.2	49.2	45.4	49.2
1976	46.0	45.4	46.9	49.7	52.5	59.0	55.5	43.3	47.0	45.1	51.1	51.9
1977	48.3	51.3	47.4	43.5	42.3	45.6	43.3	42.6	48.3	48.4	47.7	52.2
1978	55.2	55.8	54.5	51.9	48.1	50.6	51.7	47.1	51.1	51.7	53.9	53.0
1979	54.5	59.1	77.6	54.2	51.0	54.7	56.5	53.8	52.2	51.4	55.9	56.4
1980	56.1	55.0	53.0	50.4	48.4	48.1	52.1	48.0	50.2	51.2	52.2	51.3
1981	63.1	55.0	56.6	64.4	55.1	57.2	52.9	48.6	52.3	51.1		

Figure 6

Passenger Traffic: April to October Cumulated

<i>FRANCE COUNTRY A</i>				
	1978	1979	1980	1981
<i>AIR FRANCE</i>	257,463	286,270	257,511	287,162
<i>AIR A</i>	291,685	319,212	313,778	318,192
<i>TOTAL</i>	549,148	605,482	571,289	605,354
<i>SHARE OF AIR FRANCE</i>	46.9	47.3	45.1	47.4
<i>FRANCE COUNTRY B</i>				
	1978	1979	1980	1981
<i>AIR FRANCE</i>	203,196	228,035	224,773	260,346
<i>OTHERS</i>	318,323	328,545	308,696	326,186
<i>TOTAL</i>	521,519	556,580	533,469	586,532
<i>SHARE OF AIR FRANCE</i>	39.0	41.0	42.1	44.4
<i>FRANCE COUNTRY C</i>				
	1978	1979	1980	1981
<i>AIR FRANCE</i>	331,635	385,181	343,390	412,048
<i>OTHERS</i>	324,570	337,955	345,978	339,551
<i>TOTAL</i>	656,205	723,136	689,368	751,599
<i>SHARE OF AIR FRANCE</i>	50.5	53.3	49.8	54.8

APPENDIX 3

Quick Presentation of the Box and Jenkins Model

Let Z_t be a process, which is a set of random variables depending on time. We will use the notation found in Box and Jenkins' book, *Time Series Analysis: Forecasting and Control*. The following two examples will contribute to an understanding of the studied processes.

1. Level Z_t depends on the previous realizations of the process: Z_{t-1}, Z_{t-2}, \dots etc. If the process is simply that, and it can be explained by a linear equation, it will be described by the formula:

$$(1) Z_t = \gamma_1 Z_{t-1} + \dots + \gamma_p Z_{t-p} + a_t$$

where p is a positive integer and $\gamma_1, \gamma_2, \dots, \gamma_p$ are parameters which require estimation.

a_t is a random variable, which can't, by nature, be forecasted. It can also be called an innovation.

The series of this random variable is supposedly a sequence of uncorrelated variables following a normal (or Gaussian) distribution. Such a process is called autoregressive of order p , $AR(p)$, by analogy with a multiple linear regression model in which the explanatory (or independent) variables are previous values of the dependent variable.

To simplify the following equations, some simple operators have to be defined. B , the backward shift operator, is defined as:

$$BZ_t = Z_{t-1}$$

so that, $B^m Z_t = Z_{t-m}$ (for m a positive integer).

If the autoregressive operator is defined as:

$$\gamma(B) = 1 - \gamma_1 B - \gamma_2 B^2 - \dots - \gamma_p B^p$$

then, the formula (1) can be written in an equivalent way:

$$(2) \gamma(B)Z_t = a_t$$

Level Z_t is defined by the innovations a_t, a_{t-1}, \dots , etc., which happened in the past. A linear expression of this process is:

$$(3) Z_t = a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \dots - \theta_q a_{t-q}$$

where q is a positive integer and $\theta_1, \theta_2, \dots, \theta_q$ are parameters which require estimation.

Thus, the successive values of Z_t appear to be a moving average of the past innovations. Such a process is called moving average of order q , $MA(q)$. If we define the moving operator as:

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \dots - \theta_q B^q$$

the formula (3) becomes:

$$(4) \quad Z_t = \theta(B)a_t$$

In reality, the studied processes are such that the actual level of Z_t is determined by both the past realizations of the process and the past innovations. Using the same notation, such a process, called an ARMA, can be written as:

$$(5) \quad \gamma(B)Z_t = \theta(B)a_t$$

Actually, under certain hypotheses, an ARMA process can be considered as a pure AR or a pure MA. Indeed, if we suppose:

$$\pi(B) = \theta^{-1}(B) \times \gamma(B) = \sum_{i=0}^{\infty} \pi_i B^i$$

and if θ^{-1} is the inverse series of θ , then formula (5) can be written as:

$\pi(B)Z_t = a_t$ which can also be written as:

$$(6) \quad Z_t = a_t - \pi_1 Z_{t-1} - \pi_2 Z_{t-2} - \dots$$

However, the part of the equation to the right of the equal sign only has a significance when the π_i tend quickly towards zero, the requirement being that the solutions to the polynomial all have a modulus greater than 1. The level Z_t of an ARMA process is then dependent on the past values Z_{t-1} , Z_{t-2} , ..., etc.

In a similar way, let γ^{-1} be the inverse series of γ and let us suppose that:

$$\Psi(B) = \gamma^{-1}(B) \times \theta(B) = \sum_{i=0}^{\infty} \Psi_i B^i$$

then, formula (5) becomes:

$$(7) \quad Z_t = \Psi(B)a_t$$

or, $Z_t = \Psi_0 a_t + \Psi_1 a_{t-1} + \dots$ ad infinitum.

Just like the π_i , the Ψ_j must satisfy certain conditions so that this expression has a significance. The first condition is that the process must be stationary which means that the mean and the variance must be constant and finite. In addition, the covariance of Z_t and Z_{t-h} must depend only on h and not on t .

The only processes that can be studied with an ARMA model must satisfy these conditions. The stationary hypothesis is very restrictive. For example, a phenomenon which manifests a growth trend, as in the case for the air traffic, is not represented by a stationary series, and no ARMA model can therefore fit it. However, there is a class of non-stationary processes which, after some transformation, can generate stationary processes. Thus, even if the variable Z_t contains a trend, it is possible that $W_t = Z_t - Z_{t-1}$ is stationary and can therefore be fitted by an ARMA model. It may be necessary to difference the series W once more to make it stationary; this can be the case if the process contains changes in trends. In general, we define the differencing operator as:

$$\nabla = 1 - B$$

the process $W_t = \nabla^d Z_t$ is the difference of order d of the process Z_t . If Z_t can be fitted by an ARMA model, we have:

$$(8) \quad \phi(B)\nabla^d Z_t = \theta(B)a_t$$

where ϕ and θ are respectively polynomials of degree p and q , all the solutions of which have a modulus strictly greater than 1 (a necessary condition for an ARMA model). Such a process Z_t is called an ARIMA (p,d,q) . (ARIMA means Auto-Regressive Integrated Moving Average).

The word *integrated* comes from the fact that to reconstitute the original series from the differenced series, one must construct its sum (integration).

Other transformations can be applied to Z_t to produce a stationary process. An example of a commonly used transformation is:

$$W_t = \nabla \text{Log } Z_t$$

A series which manifested a constant growth rate would be made stationary with this transformation.

Seasonal models

A seasonal phenomenon can't be studied with the process defined by the general formula (8).

Indeed, if the seasonal movement isn't extremely simple (represented by a sinusoid), there is no obvious transformation that can produce a stationary process. This doesn't mean that model identification is impossible but rather that it would lead to high values for p,d,q . There would then be a large number of parameters to be estimated, which would be a problem if the studied chronology did not include many observations. That's why Box and Jenkins proposed a model based on the following simplifying hypothesis:

Each of the 12 series (for monthly series) associated with each month of the year can supposedly be identified as a single model:

$$(9) \quad \Phi(B^{12})\nabla_{12}^D Z_t = \theta(B^{12})\alpha_t \quad 12$$

where:

$$\nabla_{12} = 1 - B^{12}$$

Φ and Θ are polynomials of order P and Q . As the months are not independent of one another, the residual series α_t can supposedly be identified by a model:

$$(10) \quad \phi(B)\nabla^d \alpha_t = \theta(B)a_t$$

By eliminating α_t using (9) and (10), we get the general formula for a seasonal model:

$$(11) \quad \phi(B)\Phi(B^{12})\nabla^d \nabla_{12}^D Z_t = \theta(B)\Theta(B^{12})a_t$$

Such a model is called a SARIMA.

Implementation

The number of parameters to be estimated can seem impressive: the orders of differencing, the degrees of the polynomials and their coefficients. In reality (see Appendix 4) this number stays limited.

The main difficulty stems from the fact that the studied chronology, appears generally (with the approximations needed for any use) as the realization of several very similar processes, but the formulations of those processes (11) can look very different. It results that a process is rarely obviously identifiable; it is necessary to estimate several models, to apply appropriate statistical tests to these estimation results, and finally to choose the fitted model. This procedure gives the method, mathematically elaborated, an empiric aspect that can only be mastered with experience.

A model having been chosen, the forecast originating at date t for the coming dates $t, t+1, t+2, \dots, t+l$ can be derived for formula (11) by setting $a_{t+l} = 0$ for $l > 0$.

In the event of an important innovation after date t , the forecasts are probably going to be far removed from reality, which is precisely what allows a quantification of that innovation. It's on this aspect of forecasting that this study rests.

It is impossible in such a short presentation to describe the methodology of the three steps which constitute the construction of an ARIMA model: identification, estimation and checking. The interested reader should consult the references at the end of this paper for a more detailed discussion.

APPENDIX 4

USED MODELS AND SIMULATIONS

The problem of strikes

Strikes are not ordinary innovations, and don't seem to register in the memory of the process (1). The problem created by the resulting aberrant points is not as important for the estimation of the parameters as it is for the perturbation that will change the forecast function. The method to eliminate this problem consists in correcting the most recent aberrations. For instance, to correct the fall of traffic resulting from the ATC strike (2) in November and December 1979, an estimation was first performed using the series taken only until October 1979. The values for November and December were then replaced with the forecasts resulting from this first estimation, which leads us to suppose that the innovation of the process would have been zero for November and December 1979. It is possible then to link several estimations; an example is the series of the traffic between France and Country C, in which a strike of the cabin crew of Air C in March 1979 strongly perturbed the distribution of traffic between the different companies.

- (1) J.P. Indjehagopian studied the series of passengers at the Paris airport and showed that the ATC strike was not followed by any compensating traffic increase. This proves that the only adjustment needed is for the period of the strike. No adjustment is necessary for the periods immediately following.
- (2) Air Traffic Control

This method is not the only one. In particular there is another method based on Box and Tiao's models (1), which was used by J.P. Indjehagopian to study the impact of the ATC strike of 1979 on a series of traffic at the Paris airport. However, on the one hand, this is not the subject of our study, and on the other, the software *ad hoc* did not exist on the I.P. Sharp system at the time of the study. Hence a simpler method was used.

The models

The model chosen for each studied series is described using the Box and Jenkins notation given in Appendix 3. We're using a series (after correction for strikes) running from January 1975 through March 1981.

- (1) These models are: $Z_t = F(p, \xi_t, t) + N_t$ where ξ_t represents the additional (deterministic) effects of an exogenous variable, and p a set of parameters. N_t is a random variable which can be modelled according to an ARIMA scheme.

Note

Every model uses the logarithm of the variables. This is necessary because, in the case of seasonal models, the amplitude of the seasonal wave is roughly proportional to the level reached by the series. The seasonal component of the logarithms of the series is

pretty stable. This aspect is only noticeable in long series; however, this logarithmic transformation is consistent with the use of deseasonalizing with multiplicative models. No other more complex transformations were used on this series because of its length (6 years). As a consequence of this short length of the series, the seasonal component of the processes always appears very simple. More exactly, the equation (9) in Appendix 3 can be written:

$$\begin{aligned} \text{Log } Z_t - \text{Log } Z_{t-1} &= (1 - \Theta B^{12})\alpha_t \\ (\text{i.e.: } \Phi &= 1, D = 1, \Theta(B^{12}) = 1 - \Theta B^{12}) \end{aligned}$$

which means that each of the 12 little annual series associated with each month arrange themselves around a log-linear trend, the errors being the result of an extra seasonal movement.

Let us note that the process identification procedure could have brought other suggestions, but as those didn't seem better, we used a systematic procedure. It results that all our care was given to the estimation of the regular component of the process—i.e., to ϕ and θ , and not to the seasonal one.

The three following pages show the models chosen for each country and for each company (9 altogether). Notation is described in Appendix 3. In addition, σ_a^2 represents an estimate of the variance of the residuals a_t . The numbers following the estimates of the parameters are the standard deviations of those estimates.

FORMULAE USED TO CREATE MODELS

France/Country A

Air France:

$$(1 - \phi_1 B - \phi_2 B^2) \nabla \nabla_{12} \text{Log } Z_t = (1 - \theta_3 B^3) \times (1 - \Theta B^{12}) a_t$$

$$\phi_1 = -0.6067 \text{ (0.1305)}$$

$$\theta_3 = 0.5682 \text{ (0.0839)}$$

$$\phi_2 = -0.6555 \text{ (0.1195)}$$

$$\Theta = 0.3086 \text{ (0.0424)}$$

$$\sigma_a^2 = 0.0121$$

Air A:

$$(1 - \phi_1 B) \nabla \nabla_{12} \text{Log } Z_t = (1 - \theta_1 B) \times (1 - \Theta B^{12}) a_t$$

$$\phi_1 = 0.4071 \text{ (0.1189)}$$

$$\theta_1 = 0.8751 \text{ (0.0703)}$$

$$\sigma_a^2 = 0.0063$$

$$\Theta = 0.7211 \text{ (0.0498)}$$

Total:

$$\nabla \nabla_{12} \text{Log } Z_t = (1 - \theta_1 B) \times (1 - \Theta B^{12}) a_t$$

$$\sigma_a^2 = 0.0044$$

$$\theta_1 = 0.7207 \text{ (0.0892)}$$

$$\Theta = 0.6822 \text{ (0.0720)}$$

Adjustments for strikes:

	<i>AIR FRANCE</i>		<i>AIR A</i>		<i>TOTAL</i>	
	<i>ACTUAL</i>	<i>ADJUSTED</i>	<i>ACTUAL</i>	<i>ADJUSTED</i>	<i>ACTUAL</i>	<i>ADJUSTED</i>
NOVEMBER 1979	29,315	42,300	33,021	47,700	62,336	90,000
DECEMBER 1979	26,446	33,000	30,347	38,000	56,793	71,000

Note: Adjusted values are not exactly issued from the model, they were modified to respect the equation:

Air France's traffic + Air A's traffic = total traffic.

France/Country B

Air France:

$$\nabla \nabla_{12} \text{Log } Z_t = (1 - \theta_1 B - \theta_2 B^2) \times (1 - \Theta B^{12}) a_t$$

$$\begin{aligned}\theta_1 &= 0.4795 \text{ (0.13)} & \Theta &= 0.3418 \text{ (0.04)} & \sigma_a^2 &= 0.0068 \\ \theta_2 &= 0.3035 \text{ (0.06)}\end{aligned}$$

Others:

$$\begin{aligned}\nabla\nabla_{12} \text{ Log } Z_t &= (1 - \theta_1 B - \theta_2 B^2) \times (1 - \Theta B^{12}) a_t \\ \theta_1 &= 0.4461 \text{ (0.13)} & \Theta &= 0.4855 \text{ (0.05)} & \sigma_a^2 &= 0.0029 \\ \theta_2 &= 0.4125 \text{ (0.07)}\end{aligned}$$

Total:

$$\begin{aligned}\nabla\nabla_{12} \text{ Log } Z_t &= (1 - \theta_1 B - \theta_2 B^2) \times (1 - \Theta B^{12}) a_t \\ \theta_1 &= 0.6203 \text{ (0.13)} & \Theta &= 0.4438 \text{ (0.04)} & \sigma_a^2 &= 0.0019 \\ \theta_2 &= 0.1740 \text{ (0.03)}\end{aligned}$$

Adjustments for strikes:

<i>AIR FRANCE</i>	<i>OTHERS</i>		<i>TOTAL</i>		<i>ACTUAL</i>	<i>ADJUSTED</i>
	<i>ACTUAL</i>	<i>ADJUSTED</i>	<i>ACTUAL</i>	<i>ADJUSTED</i>		
<i>NOVEMBER 1979</i>	18,336	31,500	31,837	42,500	50,173	74,000
<i>DECEMBER 1979</i>	24,105	30,800	34,097	40,000	58,202	70,800

France/Country C

Air France:

$$\begin{aligned}\nabla\nabla_{12} \text{ Log } Z_t &= (1 - \theta_1 B - \theta_3 B^3) \times (1 - \Theta B^{12}) a_t \\ \theta_1 &= 0.5693 \text{ (0.1151)} & \Theta &= 0.3575 \text{ (0.0492)} & \sigma_a^2 &= 0.0101 \\ \theta_3 &= 0.1373 \text{ (0.0242)}\end{aligned}$$

Others:

$$\begin{aligned}(1 - \phi_1 B) \nabla\nabla_{12} \text{ Log } Z_t &= (1 - \theta_1 B) \times (1 - \Theta B^{12}) a_t \\ \theta_1 &= 0.3119 \text{ (0.1261)} & \Theta &= 0.5754 \text{ (0.0346)} & \sigma_a^2 &= 0.0116 \\ \theta_1 &= 0.9092 \text{ (0.0573)}\end{aligned}$$

Total:

$$\begin{aligned}(1 - \phi_1 B) \nabla\nabla_{12} \text{ Log } Z_t &= (1 - \Theta B^{12}) a_t \\ \theta_1 &= -0.5506 \text{ (0.1077)} & \Theta &= 0.2498 \text{ (0.0450)} & \sigma_a^2 &= 0.0032\end{aligned}$$

Adjustments for strikes:

	AIR FRANCE		OTHERS		TOTAL	
	ACTUAL	ADJUSTED	ACTUAL	ADJUSTED	ACTUAL	ADJUSTED
MARCH 1979	64,509	51,700	18,288	42,600	82,797	94,300
NOVEMBER 1979	41,669	49,500	32,807	41,800	74,476	91,300
DECEMBER 1979	37,568	43,550	28,809	33,850	66,377	77,400

Note: For Air C (Others) and total traffic series, the specified model is different before and after adjustments for strikes. It does not mean that the process becomes very different; indeed, similar processes can be represented by different kinds of ARIMA models (see Appendix 3).

Only the models which were obtained after adjustments used for simulation are shown above.

Simulation of the traffic for the summer of 1981

Once the model has been estimated, the forecast isn't a problem. Let us consider the simplest model, which is the one for Air France and Air A. The equation becomes:

$$Z_t = Z_{t-1} + Z_{t-12} + Z_{t-13} + a_t - \theta_1 a_{t-1} - \theta a_{t-12} + \theta_1 \theta a_{t-12}$$

If we suppose the process known until date t , then it is clear that an estimate for Z_{t+1} can be found by writing the equation replacing t by $t+1$ and supposing $a_{t+1} = 0$ (0 innovation).... and so on for later dates.

The main difficulty lies in the publication of the results. This is because the forecasts of the total traffic of Air France and of its competitors is slightly different from the total of the forecasts. The difference is quite negligible if we consider an average over several months. To present consistent results, which means to respect the equation:

$$\text{Air France traffic} + \text{competitors' traffic} = \text{total traffic}$$

The forecasts for Air France and its competitors were adjusted so that their sum is equal to the total, still respecting the proportions between Air France and its competitors. Let us note that the models on the total traffic seem better (compare the values for σ_a^2). This is understandable, as the division of the traffic between several companies includes various factors linked to the characteristics of a particular company's offerings: frequency, schedule, etc. A good model is therefore harder to obtain.

MANAGING APL PROJECTS

Roy A. Sykes, Jr.
Vice-President
STSC, Inc.
Woodland Hills, California

Abstract

One distinguishing characteristic of most small APL projects is their success in spite of the lack of visible management. Nonetheless, various forms of informal management are in fact at work. In contrast, large APL projects sometimes fail to be completed successfully precisely because they rely on these same informal management methods in inappropriate circumstances.

This paper discusses the advantages of applying APL to smaller efforts, the contrasts between small and large projects, and mechanisms whereby the same traits that make APL successful in small projects can be scaled up to large projects. Addressed mainly to technical managers, the paper examines issues of standardized software, programmer motivation, project management, and technical leadership.

Introduction

Some projects succeed, others fail, while some just seem to muddle along. This state of affairs exists in all fields of human endeavor, including programming—yes, even APL programming. Many times the boundaries between project success and failure are fuzzy. Is the four-month-late project a failure if it performs far beyond expectations? Is the project that's delivered on time, per specification, and in budget a success if two programmers resign after its completion? Is the never-ending amorphous growth of a large, ill-designed system cause for alarm if users continue to get the results they need? In order to discuss the subject of successful project management, we must first define what a project is, then decide what successful completion entails, and finally chart the paths to our objective.

What's a project?

For our purposes, an APL project is like an algorithm—it has input (the problem), output (the solution), and a finite series of unambiguously defined steps leading to the solution. The problem may be trivial, straightforward, or complex. Similarly, the solution may be a program, some files and a workspace of programs, or a complex system comprising many files and workspaces. However, in spite of the variety of

problems and solutions, the projects themselves seem to have the same steps to completion:

- specification—system analysis, assessing needs, proposing solutions
- design—conceiving architecture, designing files, workspaces, function calls, etc.
- programming—developing algorithms, coding
- testing—debugging, quality assurance, feedback
- turnover—documentation, training, acceptance

Various authors prefer other terms, the wording often is embellished or intermediate stages added, and steps are sometimes combined, repeated, or seemingly elided. Nonetheless, these represent a useful minimal set for this discussion.

We will further limit our definition of a project to one whose client (originator and acceptor) is other than the individuals completing the effort. This distinction is important. A programmer called upon to do a certain project may divide it up into various tasks. If he alone completes them, they are just that—tasks. If he delegates some of the tasks to others, those tasks become projects to the delegates, and he becomes the client. Essentially, one project can have subprojects only if several people are involved in the project.

We will not consider maintenance to be a project, but rather an activity. Ongoing maintenance involves unknown problems and (at least to the maintainer) a seemingly infinite series of ill-defined steps to get things working again (the solution). Likewise, implementing one's own pet project—no matter how extensive—will not be considered a project for our purposes. Nor will we consider a project one of those never-ending amorphous enhancement efforts. Projects have a beginning, an end, and a client.

Two kinds of projects

There are big and small projects, and this is an area in which APL differs from other languages. The rapidity of development and functional capabilities of small APL projects vastly exceeds that of equivalent projects in languages like BASIC, COBOL, FORTRAN, PL/1, or PASCAL. APL's advantage tends to diminish abruptly, however, once projects reach a certain size, which we'll call "large".

Put in terms of human resources, small projects are those completed by one person (only) in less than six months. Large projects are those completed by two or more people in two or more months. There is indeed a gap in one sense, and an overlap in another. Three people working on one project for one month has characteristics of both small and large projects. Such projects are also pertinently rare in APL. However, a five man-month project is distinctly large if completed by two people, but small if completed by one.

Notice that the difference between large and small projects is not defined in terms of the complexity or scope of the project. In fact, small projects sometimes result in more powerful and useful software than large projects, particularly in APL. Projects in APL are subject to the same mythical man-week effect (paraphrased from Fred Brooks) as

those in other languages. Asking a programmer to complete a project in half the time by assigning him another programmer is like expecting two pacers to draw a sulky twice as fast as one.

Measures of success

Successful projects are generally accompanied by happiness and satisfaction on the part of the clients, the developers, and the managers. Unsuccessful projects breed a variety of less pleasant feelings. Subjectively then, we usually know when a project has been successful. There are definite exceptions and always degrees of success. The client may be satisfied, but the programmers and their manager may be pulling their hair out. The people assigned to the project may feel they've met all criteria, yet the client may be displeased. A programmer can "burn out" on an otherwise successful project.

Some organizations attempt to quantify project success using objective measures such as "within 10% of budget" or "no more than 15 days late". The flaw in these metrics is their ignorance of the human element—after all, humans assign, deliver, and accept projects. A project delivered early and under budget that does not satisfy the (perhaps unspoken) needs of the client cannot be considered successful. Nonetheless, these measures, applied judiciously, can point out areas for improvement. In the example just cited, perhaps the process of assessing needs and proposing solutions needs refinement.

Furthermore, few projects are so well-defined and static that the objective measures for their success remain constant. To the contrary, especially on large or complex projects the ultimate target often is moving. Specifications change mid-stream, system performance changes, timetables shift, and personnel become unavailable. All participants in the project must be apprised of these changes, and any objective success metrics must be adjusted. Ultimately, however, success is not measured in man-days or dollars, but by the satisfaction of the participants.

Don Scheer, senior APL Applications Analyst with the OPTIMATION Group of STSC, has suggested the final stage of a client (external) project be an interview with the client, the programmers, and the managers involved. We are currently refining that process and developing an objective rating system to measure the success of consulting projects. We emphasize that this is not just a client satisfaction survey, nor simply an internal comparison to original estimates. It does include these factors, but just as importantly, it includes assessments of management and project leader perceptions, and the feelings and satisfaction of all individuals involved in the project. We want to know if a project was successful, or why it was not.

Small is beautiful

Small projects provide much of the grist for APL's reputation for productivity. The war stories abound: lone APL programmer completes in five weeks what a four-person COBOL team took three months to write; 9,000-line FORTRAN system replaced by 1,500-line APL system; analysis system written in one week by APL programmer. These testimonies all reflect successful projects; moreover, each project was small by our definition. Only one programmer was involved in each case, and the longest duration was 4.5 months. What is it about small APL projects that typically make them so successful?

The primary success factor is that only one person works on the project from start to finish. In each case, the programmer's manager was directly involved only in soliciting the business (contract negotiation, preliminary specification, and estimation), conducting periodic reviews with the programmer and client, giving occasional technical advice, and final turnover. Most of the client contact and all of the technical work was done by the programmer. Essentially, the programmer managed the project and his manager managed the environment.

The programmer in such single-person projects performs many roles, frequently switching among them. He manages and executes customer relations, systems analysis, design, programming, quality assurance, and documentation. This mode of operation is highly motivational and effective in spite of the time lost in switching roles. Competent individuals usually rise to responsibilities for which they feel they have full authority. Furthermore, the creative challenge of integrating many aspects into a cohesive system seems to inspire APL programmers, who generally tend to be creative and bright. Rarely do APL development organizations classify their personnel as rigorously or hierarchically as, say, COBOL-based organizations. While the latter have user specialists, systems analysts, designers, programmers, project managers, technical writers, and a slew of other titles, APL organizations often have just programmer/analysts and technical writers.

When one person is developing an entire project, that person is executing "in core" almost all the time, and consequently requires relatively little management (operating system resources). He does not need to explain often complex concepts to other people, and little information is lost in external communication channels. The fact that he is using APL accrues many associated benefits. Design is simplified and strengthened by the conceptual tools APL provides. Programming and debugging time is reduced. Prototyping and interactive design become viable techniques. The net result is that fairly extensive projects can be completed by one APL person in a short time. In other languages, either the timeframe for one person would be unacceptably long, or the benefits of a single person are lost. Big systems can be small projects in APL.

The ability to remember and manage all aspects of a project "in one's head" is limited, of course, and depends on the individual and the duration and complexity of the project. We find that projects of up to five or six months can be completed effectively by a single competent individual, assuming the time is available. Beyond that time, it pays to have other people, often specialists, aid in areas like project management, documentation, programming, and customer relations. However, as soon as additional people are assigned to a project, a whole set of problems arise.

The large project

Some APL projects involve many individuals or extend for many months. They may be of modest scope but severely limited duration, or simply of great complexity. Attempting to handle them identically to small projects often results in failure. Again the war stories: three-person APL team promises four months, delivers in nine; \$40,000 project not completed, \$25,000 refunded; new system remains incomplete and error-prone five months after delivery. Why are these failures more prevalent than in small projects, and how can we make large projects successful?

One reason for failure is simply lack of effective project management. A lead programmer might be thrust into the position of project manager with little or no experience, and less help from his manager. He may be a superb programmer/analyst, but have little interest or aptitude for managing other people. Even experienced project managers sometimes don't pay adequate attention to coordinating resources, preferring to "manage by milestone" rather than manage through people. Some managers try to achieve a consensus on every design decision from all the programmers, with consequent duplication of effort and wasted time. Others accept a single person's design with no review or feedback from others knowledgeable in the requirements, often resulting in resentment, major "holes" in the design, or later reprogramming. A project manager has to manage.

Project failure sometimes occurs because many APL programmers are generalists who do not have the formal background, training, experience, or skills necessary to adequately perform other assigned tasks. For example, a comprehensive accounting system needs a designer knowledgeable in finance. A system for world-wide data collection and reporting needs an analyst who can view the entire project as an integrated system, not just an APL computer implementation. A system used by non-technical people needs to be documented by a skilled writer, not an APL programmer who writes jargonese. As projects get larger, the degree of specialization required increases. While the analysis and design skills of an APL programmer may be adequate for a small project, they can be dangerous ("a little knowledge ...") in a large project. Large projects require a disciplined and professional approach.

APL encourages conceptual overviews of data processing. It eliminates many—but not all—of the minutiae of programming and design. APL simplifies development. Yet these strengths often lead to oversimplification of the intricacies of a large system. The result often is naive and optimistic estimates of the time and people required to complete a large project. For example, the designer may properly estimate the time to develop the primary database and programs, but may overlook or oversimplify the myriad supporting details of auxiliary tables and their maintenance (add, delete, review, change), initialization or conversion routines, or adequate documentation. In small systems, these details are a more significant portion of the overall effort, and are less likely to be overlooked.

Making large projects work

How can we increase the probability of success in large projects? There are two complementary aspects. One is the environment in which the project team exists, the other is the makeup of the team itself. The organization from which project workers are drawn should be cohesive, supportive, and adherent to certain standards. The project team then must be structured properly, managed well, and insulated from distraction.

The organization

There are several characteristics vital to an effective APL development organization. These background factors enable the individuals selected for a specific large project to have certain mutual understandings in advance of project commencement. Attempts to develop these characteristics "on the fly" during the project dilute efforts, waste time, and are seldom completely successful. The organization's climate must be conducive to ad hoc project demands.

Accepted standards

Everyone in the organization should adhere to certain standards for estimation, design, programming, utilities, and documentation. Common units of measure (pages, man-days, dollars, lines of code, etc.) should be adopted. Unique or idiosyncratic file structures and programming techniques should be discouraged. A universal set of powerful, efficient, and well-documented utility functions and development tools must be available and used by everyone. Stylistic guidelines should be promulgated. Minimum standards of programmer and user documentation must be followed.

These standards should not be imposed on the people in an organization. Rather, they should represent a consensus of established and desirable practice. Nor must the standards remain static; new ideas should be encouraged and the best integrated gradually through widespread acceptance. Finally, everyone should recognize that the standards are not unvaryingly absolute. There are circumstances under which even the most robust standard may, and indeed should, be violated. The justification must be very strong and all involved should agree. Essentially, we depend on the professionalism and expertise of the individuals involved.

Recognized specialties

Each individual in a development organization has a job title. The analysts analyze, the programmers program, and the documenters document. But each individual has his own particular set of interests and specialties; these distinctions should be known to the others in the group. One person may be particularly strong in statistics, another in database management. Some programmers code rapidly, but require more debugging time; others code slowly and methodically, producing highly reliable results. Some people react better to pressure or frequent interruptions than others. A strong organization encourages individuals to develop their own specialties, then exploits those traits to the benefit of the individual, the organization, and its projects.

These characteristics are not necessarily related to individual job descriptions. For example, a programmer may be a natural leader to whom others look for guidance. A designer may have a particularly lucid writing style which he, occasionally, likes to exercise. A technical writer could deal unusually well with clients. A manager might have some extraordinary but latent technical specialty. If everyone is aware of these capabilities, they can be used effectively in projects, and provide a creative outlet for the employees.

Supportive management

A software development organization relies on people more than, say, a manufacturing organization. The productive facilities are humans rather than machines. The largest expenses are salaries rather than raw materials or capital equipment. Clients have an unusually large say in the resulting product. Therefore, managers must be highly oriented to people. Of course, all managers work through other people, but managers of programming groups must be especially sensitive to the needs of their subordinates.

Managers should have a background in design and programming. The manager need not be the technical leader of the group, but he should comprehend and be able to contribute to technical issues. He should understand the creative process, and the motivations of those through whom he works.

If a manager is supportive, his subordinates will feel free to communicate problems to him when they arise. He will attempt to insulate them from administrative tasks and irrelevant interruptions. He will be able to mediate technical or personal disputes. He will be able to allocate projects that best use the resources of his group. He will be able to motivate his people in a variety of ways: technical challenge, career growth, compensation, travel, and perquisites. Most importantly, he will be a team builder.

The project manager

Let us assume the project team has been selected from an effective organization. What are the dynamics which affect project success? The most important is project management. We know the large project has to be effectively broken into several smaller projects (each potentially with its own team of one or more people), and that these subprojects may themselves have to be broken into yet smaller projects. The project manager has to be able to allocate these subprojects properly, and to help the managers of the subprojects allocate resources.

Ideally, the large project is ultimately divided into many small (using our definition of the word) projects for which single individuals are responsible. Since it is likely that managers of the intermediate projects are not well-versed in project management, the main project manager has to become involved in subprojects. Essentially, these ad hoc project managers are really team leaders, and they depend on the primary manager for project management. A strict military (hierarchical) organization does not work effectively in projects; a matrix structure is more appropriate.

The technical leader

One person, not necessarily the project manager, should be the technical leader of the project. He should already be recognized as having the strongest technical talent in the team. He is the individual to whom all turn for design issues, advice on implementation, and technical guidance. He may delegate decision-making to others, but he becomes the focus for those decisions, and disseminating the results. In most projects, the technical leader will also have other duties such as analysis, design, or programming. But his informal role is that of technical coordinator.

The administrator

Someone has to track the project, expenses, milestones, and so forth. Meetings with the client, telephone calls, reports, and budgets have to be handled and arranged. The project workers should be insulated as well as possible from these administrative headaches. Often the project manager will assume this role. The point is to let the programmers program, not inundate them with meetings and paperwork.

The technical writer

This person assures that the system is adequately documented, both for programmers and users. He therefore becomes a focus for information from all aspects of the project. While the formal part of the technical writer's job (documentation) is most important, the informal role he plays is also important. He can alert others to possible inconsistencies in the system. In the course of producing examples for publication, technical writers often play the role of quality assurers. Because they deal informally with most of the project team, they can often recognize problems early and bring them to the attention of management. Usually, technical writers are held in high esteem by programmers—who typically hate to document.

The team

The four key roles (project manager, technical leader, administrator, technical writer) may be played by four different people, or (often in APL projects) by one or two people. The rest of the team usually consists solely of APL programmer/analysts, some of whom may also play one of the key roles. If the project is large enough, some of the senior people may also be subproject leaders, often assisted by the project manager.

Each project should be clearly defined and specified in writing. The specification for the overall project may run dozens or hundreds of pages; for a subproject it may only be a page or two. Furthermore each project should be estimated, and the time estimates reviewed and updated periodically to reflect actual progress. Openness and honesty must be encouraged, and the technical leader should review these estimates. If slippage occurs, the project leader must be informed so he can take appropriate action (notify the client, add resources, etc.)

Conclusion

Part of the success of APL is that what would be large, multi-person projects in other languages often become small, one-person projects in APL. A single person typically can manage his own resources more effectively than a manager can manage the resources of several people.

Large projects in APL require that the underlying organization be supportive, and that the project team have effective management, technical leadership, administration, and documentation support. The internal roles in the single-person project must be formalized in a multi-person project. Although the imposition of formal structure (which is no different from that of project teams in other languages) diminishes the advantages of APL, the reduction in number of people still gives projects written in APL a substantial advantage.

MONITORING THE U.S. ECONOMY USING CITIBANK DATA BASES

Jonathan Sheer
Assistant Manager, CITIBASE
Economics Department, Citibank, N.A.
New York, New York

Ladies and gentlemen, it's both an honor and a privilege to address this APL Users Meeting. My assignment this morning is to explain how managerial decisions can be improved by monitoring the United States economy with information from Citibank's economic databases. But before turning to the substance of my remarks, I must begin by saying something about what it is that we at CITIBASE do.

About U.S. data and CITIBASE databases

Citibank's economic databases, CITIBASE and CITIBASE-WEEKLY are reporting mechanisms. Almost every day new data are released by the various agencies of the U.S. Federal Government. These data are collected by our staff, organized, checked for consistency, and entered into the databases. The new or revised data is then transmitted to I.P. Sharp where it is updated within minutes.

The United States, unlike Canada, does not have a central statistical office. In fact, CITIBASE collects data from more than twenty separate issuing agencies. Data are released on a relatively regular cycle: Producer Prices in the first week of the month, Consumer Prices and Gross National Product in the third week, etc. Note that although the Gross National Product figures are quarterly, they are revised every month.

What's important to remember is that economic data are often subject to revision. These revisions may affect only the last observation, or they may affect the whole series of weekly, monthly or quarterly observations. Quite often it is somewhere in between. Take GNP as an example. Each quarterly figure is revised twice after the preliminary release. Then, once a year, a "benchmark revision" changes five years of GNP data.

Revisions are made for a variety of reasons—additional information, new methodology, new seasonal factors, a shifting of the base year, or a change in concept underlying a particular time series. Revisions often strike without warning—making it very important that you check the consistency of the historical data, especially if you intend to match some new data up to a job you have worked on in the past. If you are not careful, revised data can result in costly errors, particularly when they are the basis for forecasts and the business plans that rest on those forecasts.

What do you do with the data that CITIBASE so meticulously prepares? The answer, in a nutshell, is that the data—the quantitative information—must be an integral part of an analytical framework. There's nothing more pernicious, nor more dangerously naive, than to suppose that "the facts will speak for themselves". Facts never speak for themselves, and people who proceed on the opposite assumption are more often than not implicitly relying on some theory, and it's usually bad theory. As John Maynard Keynes put it nearly fifty years ago: "...the ideas of economists and political philosophers, both when they are right and when they are wrong, are more powerful than is commonly understood. Indeed the world is ruled by little else. Practical men, who believe themselves to be quite exempt from any intellectual influences, are usually the slaves of some defunct economist. Madmen in authority, who hear voices in the air are distilling their frenzy from some academic scribbler of a few years back."

The analytical framework of which I speak can be viewed on several levels. On the narrow or more technical end of the spectrum, there are specific-quantitative forecasts—either of the behavior of entire economies or forecasts that are more finely targeted to specific industries, markets or individual product lines. The broader end of the analytical framework has to do with a sense of economic direction—questions about where the United States and other technologically advanced economies have been and where they are going. It's to this big picture that we must now turn.

The inflation phase

When you look at United States GNP—real GNP, measured in constant prices—you're immediately struck by the fact that there has been virtually no net change—no growth of real GNP—since 1979. The Citibank forecast shows this pattern of stagnation ending soon—perhaps with solid gains by the end of this calendar year. But the delayed recovery is not expected to be spectacular. And there remains the unanswered question—why the three years of economic stagnation, not only in the United States but throughout the technologically advanced countries of the world?

The brief answer—which I shall hasten to expand—is that the stagnation is one of the more painful manifestations of a worldwide deflation. Let me quickly sketch the sequence of events as we view it at Citibank.

After nearly five years of price stability, the United States—beginning with the Tonkin Gulf resolution in 1964—embarked on a monetary expansion—an acceleration in the growth of its money supply—that was destined to touch off a wave of inflation. The shift to an inflationary monetary policy was principally occasioned by the war in Vietnam. But that military conflict also coincided with President Johnson's "war on poverty", and there was a massive increase in federal spending for programs designed to enhance welfare and transfer income.

As inflation accelerated in the United States, there were rather prompt international repercussions. The U.S. dollar was the centerpiece of a system of fixed exchange rates among the advanced countries of the world. And when its purchasing power shrank rapidly, dollar exchange rates came under strong downward pressure. Other countries—West Germany was the most striking case—attempted to prevent the fall of the dollar by intervening in the foreign exchange markets. The Deutschesbank made large purchases of dollars from West German holders, and in carrying out such support operations, it sharply accelerated the growth of the West German money stock. And so U.S. inflation was exported.

Pressures mounted, not only on dollar exchange rates but on the ability of the United States to maintain the gold convertability of the dollar among central banks. In August of 1971, Washington threw in the towel. President Nixon closed the gold window and the U.S. dollar was floated. With the disintegration of the system of fixed exchange rates, rates of money supply growth continued to accelerate. And upward pressure on price levels was intensified by the OPEC price hikes, beginning in the winter of 1973-74.

Governments around the world attempted to slow the inflation through measures of monetary restraint. And the result of those efforts were the very severe recessions that gripped most advanced countries in 1973-75. But fear of unemployment—especially in the United States—caused most governments to be less than resolute in their efforts to damp inflation. After slowing money growth in 1974 and 1975, most countries reversed their restrictive monetary policies and reflat. The result was a recovery accompanied by yet another wave of inflation that peaked in 1980. But that second inflationary wave was unique in that its peak coincided with the more resolute determination of government—especially in the United States—to achieve a measure of price stability. And that brings us to the deflation phase.

It's hard to pinpoint the fine line that divides that period of inflationary monetary policy and the turn toward deflation. One astute observer of the recent policy scene in the United States points to a 1978 statement by Treasury Secretary Blumenthal in the Carter Administration and Charles L. Schultze, Chairman of the Council of Economic Advisers, to the effect that priority should be given to fighting inflation, not unemployment. Others point to the Federal Reserve pronouncement of November 1979 that monetary policy would be principally guided by money growth targets rather than a concern for levels of short and long term interest rates. But everyone agrees that the process was clearly hastened by the ascension to political power of Margaret Thatcher in the United Kingdom and Ronald Reagan in the United States.

The Reagan Administration left absolutely no doubt about where it stood in the battle against inflation. For the first time—at least in the modern era—the President clearly pinned the blame for inflation on excessive monetary growth. And his Administration broke new ground by publicly announcing that the rate of monetary growth should be reduced by 50% over a 5-year period. And then there are the dramatic and, on the whole, successful efforts to slow the growth in federal spending, a struggle that is still far from over.

U.S. inflation, as measured by the GNP deflator, reached a peak of 10.7% in the fourth quarter of 1980 and subsequently fell to 3.5% in the second quarter of this year. That's an impressive drop in inflation, but it would be even more welcome had it been not accomplished at such a high cost—costs measured by unemployment and contractions of the real output of goods and services. Some economists—especially the extreme proponents of what's called the "rational expectations" thesis—argue that peoples' expectations of future prices can be changed in such a smooth way that inflation can be lowered without depressing real output and employment. But there's surely no historical evidence that supports such a happy scenario. What actually happens is this:

- 1) High inflation causes the central banking authorities to slow the growth of the money stock. The slowing of money growth means less cash in the hands of households and business—lower balances than they wish to hold in the light of inflation, levels of interest rates and the like. To increase their cash balances—to bring them up to desired levels—businesses and households then cut back on spending, and the result is a decline in the real output of goods and services and a rise of unemployment.

- 2) Far from adjusting instantly, expectations of future inflation moved sluggishly. People who had watched prices rise since 1964—with a few slowdowns in recessions—were hardly anticipating the sort of drop that actually occurred since 1980. And small wonder. After a time, both households and businesses develop an interest in the perpetuation of inflation. No head of a household is cheered by the news that existing home prices are falling at a time where mortgage rates remain sky high. Nor are falling product prices likely to cause euphoria in board rooms, especially those of petroleum companies that are heavily invested in high-cost explorations.

Lags and surprises are what tend to make the process of deflation painful. All deflations for which we have historical evidence—Britain after the Napoleonic Wars, the United States after the Civil War, and both countries after World War I have been disruptive, and current deflation—while in some ways unique—is no exception to that basic rule.

And one of the most painful lags of all was the failure of U.S. interest rates to fall promptly when the economy skidded into recession. We still don't fully understand why it was that interest rates remained so high for so long—and in fact still remain very high in "real" terms, that is after subtracting the expected rate of inflation from nominal or actual rates of interest. But much of the mystery can be dispelled by looking at the problem from the lenders' perspective and taking account of both inflation expectations and uncertainty.

Lenders in all places and at all times have demanded and gotten premiums—extra high rates of interest—when it was apparent that the principal of their loans would be repaid in currency of sharply diminished purchasing power. But while that proposition is axiomatic, there are enormous problems created by uncertainty. Who knew in 1964 or 1965 that inflation would run so high for the next 18 years? It's this lack of foresight that explains why interest rates—especially long term rates—were slow to rise, and once elevated are so slow to fall. Many people believe that the current drop in inflation is little more than a temporary dip, a cyclical drop that will soon be ended by another burst of inflationary monetary policy. So there's an understandable stickiness or inertia of the public's expectations.

Special factors relating to the conduct of U.S. monetary policy also played a pernicious role in holding rates up so high. Portfolio managers, who suffered large losses on bonds because of the continued failure of the government to damp inflation, gave up entirely on long-term issues and focused their attention exclusively on the short end of the market. And traders of short term securities kept an increasingly watchful eye on the weekly gyrations of the money supply. They reasoned that any outsized increase in the money stock—growth above the Fed's announced money targets—would soon be followed by a corrective action, a rise in the interbank, "federal funds" rate that could cause them to suffer losses on their inventories of government securities. So whenever they anticipated a large increase in the money stock, they reflexively dropped their bids for T-bills and notes, thus pushing up yields.

This pattern of market behavior exerted a stern discipline over the Federal Reserve authorities, forcing them to stay within their targeted money growth bands. But it was also a mixed blessing, inasmuch as the Fed was severely constrained in how far it could go in accelerating money growth, and there were times—especially in this year—when speedups seemed appropriate.

This impasse was broken in late July and in August, not only by the weak demand for credit, but—more significantly I think—by the greater willingness of lenders to supply funds at lower rates. But the problems of the money markets are not over. Rates backed up in September and will probably move slowly downward in the future, but only as the markets become more firmly convinced that the Administration and the Federal Reserve remain committed to their war on inflation. Doubts on this score still cloud the near and intermediate term outlooks.

Outside the financial sector, the impacts of deflation are painfully evident. Failure and liabilities of business failures—have hit new highs in the United States. And I hardly need to remind this audience of the difficulties—especially the concern for the debts of the less developed countries—outside the United States. But bear in mind that the international monetary system has withstood very sharp shocks in the past—the 1971 episode and the recycling of OPEC surpluses—and without sounding like Pollyanna, I'd say that it will survive these current troubles which will ease once we begin to see some signs of a recovery in global economic activity.

Three years of stagnation have taken their toll on U.S. profits. Real profits—adjusted for inflation—fell by 34% in this recession, far more steeply than the average. And profits margins have been declining since 1979.

I have spent much of your time talking about the pain of deflation, and I don't want to leave without some glimmer of a light at the end of what has surely been a long tunnel. So here is a capsule version of Citibank's most recent forecast, not only for the near future, but throughout 1987. Needless to say our product warranty does not extend to the accuracy of the forecasted numbers.

**SUMMARY FORECAST FOR THE U.S.
1982 - 1987**

	NOMINAL GNP (US\$ BILLIONS)	REAL GNP (1972 US\$ BILLIONS)	DEFLATOR 1972=100	UNEMPLOYMENT RATE
1982				
1ST QUARTER (A)	2995.5	1470.7	203.68	8.8
2ND QUARTER (A)	3041.2	1475.3	206.14	9.5
3RD QUARTER (F)	3107.4	1481.8	209.70	9.7
4TH QUARTER (F)	3190.4	1497.6	213.03	9.4
1983 (F)	3399.4	1541.0	220.56	8.7
1984 (F)	3722.6	1600.3	232.57	8.1
1985 (F)	4065.5	1658.6	245.07	7.7
1986 (F)	4423.5	1715.0	257.89	7.3
1987 (F)	4806.0	1773.2	270.99	6.9

(A) ACTUALS
(F) FORECAST

SOURCE: CITIBANK, N.A., ECONOMICS DEPARTMENT

